Basic Computing

The TRS-80 User Journal

The Compatibility Game

Software Protection

Taking Software to Court

Wordstar, PFS File, CoCo VIP Writer Reviews

Writing for a Compiler

Model III/4 to 100 Conversion Program

Using Model 4 JCL's

Apple to TRS-80 Conversions

0 60959 03
318585

$3 per copy / March, 1984

The BEST goes one step beyond!

Thousands of you have already experienced the speed and power of the DOSPLUS* IV disk operating system for the TRS-80** Model 4 microcomputer. Now Micro-Systems Software introduces: DOSPLUS IV/A.

DOSPLUS IV/A is the same great disk operating system as the DOSPLUS IV already released, but with one big PLUS! DOSPLUS IV/A includes the M-ZAL*** Disk Editor/Macro-Assembler. The best disk operating system for the Model 4 combines with the best editor/assembler to bring you all the development software you should ever need for the Model 4 at one low price.

DOSPLUS IV retailed for $149.95, but DOSPLUS IV/A will retail for only $169.95! The greatest software combination imaginable for only a $20 increase in price! No-one should be without this package! Existing DOSPLUS IV owners can upgrade their packages for only $25.00.

Micro-Systems Software's DOSPLUS IV brings unmatched speed and power to the Model 4, and now MSS brings the TRUE MODEL 4 EDITION of M-ZAL, (a regular $149.00 value) the world's best machine language development system. Now there is NO other choice when seeking the best Model 4 operating system...

Dealer inquiries invited.

Trademarks:
* "DOSPLUS" and "6.0 PLUS" are trademarks of Micro-Systems Software Inc.
** "TRS-80" and "TRSDOS" are trademarks of Tandy Corporation
*** "M-ZAL" is a trademark of Computer Applications Unlimited

MICRO-SYSTEMS SOFTWARE, INC.
4301-18 Oak Circle, Boca Raton, Florida 33431, Telephone: (305) 983-3390
Orders Only 1-800-327-8724

programmer s

READ THIS...

* WHO TO SELL YOUR PROGRAMS TO
* THOUSANDS OF COMPANY NAMES AND ADDRESSES, WITH DETAILED LISTINGS SHOWING:
  (1) WHAT PROGRAMS PUBLISHERS ARE LOOKING FOR
  (2) HOW THEY WANT YOU TO SUBMIT YOUR PROGRAM
  (3) HOW MUCH THEY PAY — AND WHEN!
* 100 CATEGORIES — FROM "ACCOUNTS RECEIVABLE" TO "GAMES" TO "VIDEO CONTROL" PROGRAMS
* HOW TO WRITE CLEAR DOCUMENTATION
* DEBUGGING TECHNIQUES

Enclose check or money order for $19.95 (No C.O.D.'s) to:

#2 **IPF Publications**
146 Country Club Lane
Pomona, NY 10970
(914) 354-5585

Order Your Copy Today!

Name .....

Address .....

City..... **State**..... **Zip**.....

 Please address correspondence to: **Basic Computing, 3838 South Warner Street, Tacoma, Washington 98409, (206)475-2219 voice, (206) 756-0448 modem.**

Authors: We constantly seek material from contributors. Send your material (double spaced, upper/lower case, please) and allow approximately 4 to 6 weeks for review. Programs must be supplied in machine-readable form on diskette or tape, clearly marked as to model and operating system. Text files may be on diskette, but please include a hard copy as well. Media will be returned if return postage is provided. Cartoons and photographs are welcome. Generous compensation will be made for non-trivial works which are accepted for publication. Basic Computing pays upon acceptance rather than on publication.

SUBSCRIPTION PRICE: U.S.: $19.97 for one year, $34.97 for two years and $49.97 for three years. **Canada and Mexico:** $29 per year, no two or three year subscriptions are offered. **All other:** $36 per year via surface mail, $78 per year via airmail. Two and three year subscriptions are not offered.

ISSN Publication #0745-9912. Basic Computing is published monthly by 80-Northwest Publishing, Inc., 3838 S. Warner St., Tacoma WA 98409-4698. Printed in the United States of America.

POSTMASTER: Please send change of address form 3579 to Basic Computing, 5615 West Cermak Road, Cicero, Illinois 60650. **Second Class postage PAID at Tacoma, WA and additional entry points.**

Our cover— 8-inch diskettes don't fit into 3½-inch drives and that is just part of the compatibility problem. This month's artwork is by Randy "Tarkas" Hoar of Centralia, WA.

Basic Computing

The TRS-80 User Journal

TRS-80 is a trademark of the Tandy Corp.

Vol. VII, No. 3 — March, 1984

ARTICLES

Basic Computing

Publisher
I. Mike Schmidt

Managing Editor
Cameron C. Brown

Technical Editor
Greg Sheppard

Associate Editors
Terry R. Dettmann
Spencer Hall
Jim Klaproth

Contributing Editors
Timothy K. Bowman
Thomas Quindry
Mark E. Renne

Advertising Coordinator
Catherine Shappee

Advertising Representatives
East of the Mississippi River
Garland Associates:
John A. Garland, Frank Surace
(617) 934-6464
West of the Mississippi River
Catherine Shappee
Advertising Coordinator
(206) 475-2219

Promotion/Circulation
Robert P. Perez

Production
Catherine D. Doud

Accounting
Helen Dalton

Editorial Secretary
Julie Bartz

Editorial

Cameron C. Brown

We have been begging for some compatibility and standardization within this new industry called microcomputing. It appears to me that at least one segment has agreed on something. Companies and their advertising agencies have decided that everything must have a lot of hype.

I have received telegrams announcing a new database program, an invitation to go on a cruise and a luncheon to learn all about a new disk drive. My desk is covered every morning with glossy folders full of four-color pictures and pages of text telling me about a new printer or a new IBM work-alike computer from a new entrant into the microcomputer race. There are phone calls telling me that a new product announcement has just been mailed to me. I am really glad that the other 200 companies that send us new product announcements each month don't do that.

A big event like Comdex is an ad agency's dream. We got boxes full of confetti and a horn to toot, a parchment scroll inviting us to a poker game, more offers of elegent cuisine and wine than even *Gourmet Magazine* could digest, invitations to nightclub acts and more. All this in the name of microcomputing! If you only knew how difficult it is to get the information to you. One writer has put it into perspective. *InfoWorld's* John Dvorak has taken to reviewing the parties, not the products. He even gives awards for the best dinner, best canapes, best dancing. My hat's off to Mr. Dvorak for treating all the hype with such a well-honed sense of humor.

What I find so amazing is that companies have been convinced that this is the way to go. Sure, the louder you yell, the better your chance to be heard. But I do wish they had something to say. The number of software product announcements that don't mention what machines the software will run on, the price or, sometimes, even the company's address, is laughable. I have engraved invitations to be sure to visit booth #2341 at 12:00 Noon, on Friday the 23rd to see a revolutionary new product demonstrated. Too bad the company didn't bother to tell me what month, city, or fair they are referring to.

My favorite has to be the invitation to coffee and danish in New York, at one of those very fancy restaurants that serves Folger's Instant Coffee, to meet the brilliant 27-year-old president of a new computer company. If they were really serious about it, plane tickets would have been included.

Three years ago we used to get glossy 8 X 10 photos of every lucky schmuck who was promoted from vice-president to executive vice-president. I always wondered if we were supposed to frame them and put them on our wall. At least those days appear to be over. Now we get glossy 8 X 10 photos of the company's new wing that has been added to their corregated metal building.

An editorial should have a moral. This month's is rather simple and Joe Friday said it best. "Tell us the facts, ma'am. Just the facts."

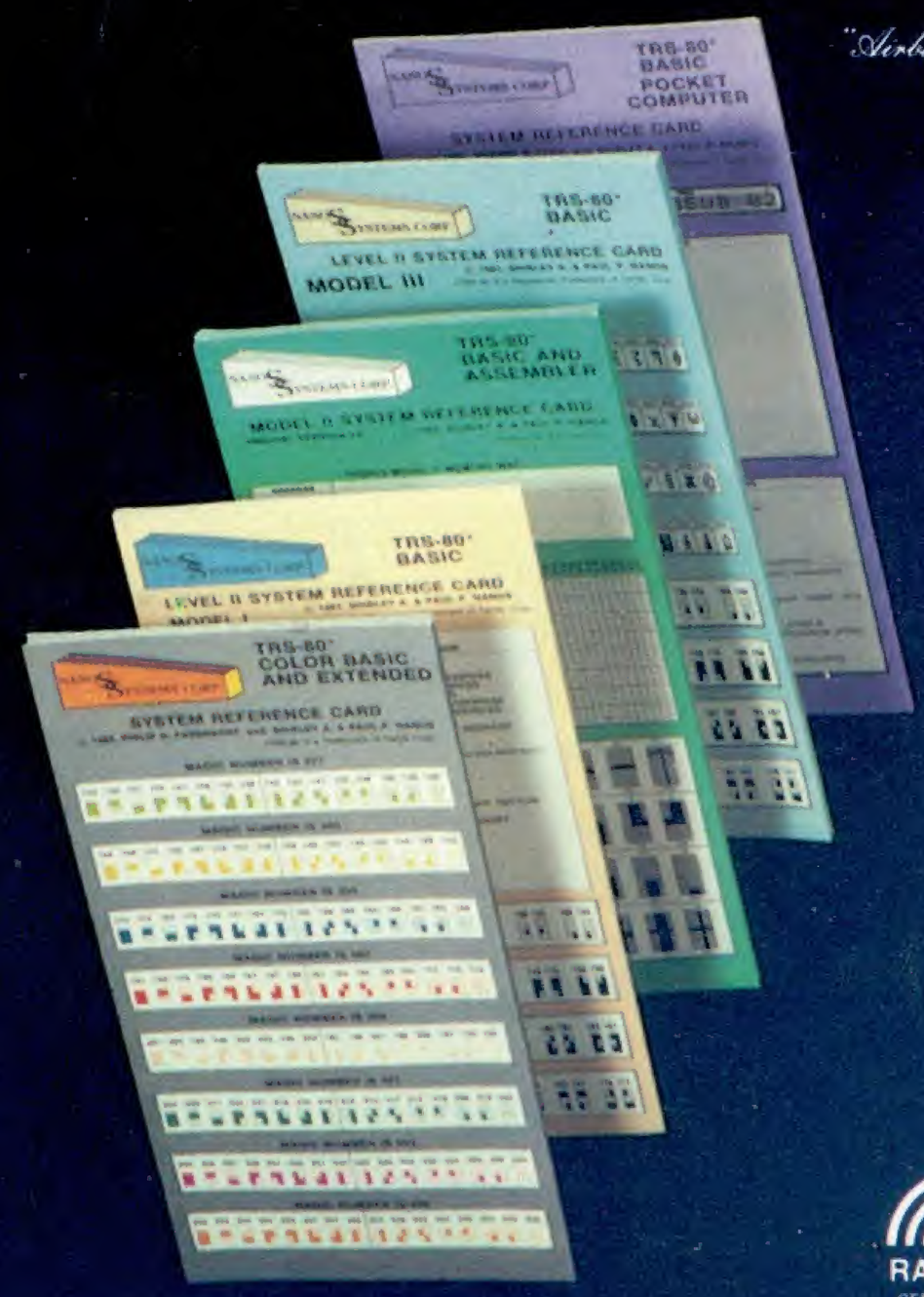
"Airborne Images"

NANOS SYSTEMS CORP.

Nanos Systems Corp.

REFERENCE CARDS

For Models I, II, III, Color, Pocket and Apple II & II Plus



RAINBOW CERTIFICATION SEAL

* TRS-80 is a Registered Trademark of Tandy Corp.
Computer picture reprinted permission Tandy Corp. APPLE is a registered trademark of Apple Computing, Inc.

Each card is a complete summary of the reference manuals and the microcomputer. Cards are two or more colors, printed on 80 pound Beckett Antique cover stock or a comparable stock, stretch-wrapped in plastic for shipping. They are accordion-foldup cards, in the same style as the traditional IBM reference cards used on the major computers for years. Fold-up size is eight and one-half by three and three-quarter inches, so they will fit easily into the shirt pocket. These cards provide a complete summary of the manuals plus many extras at your fingertips.

MODEL I

BASIC: Buff & Blue
5 Panels, 10 Pages
(For the Classroom)
Memory Map.
Easy Graphics.
Basic Statements.
Basic Functions.
Basic Facts.
Special Characters.
Basic Commands.
Edit Subcommands.
PRINT USING Examples.
Message & Codes.
Reserved Words.
Special Keys.
Ascii Character Chart, with Space Compression Codes.
Control Codes.
Basic Internal Codes.
Hex/Dec Conversion Chart.
Screen Line Layout.

BASIC & ASSEMBLER: Buff
8 Panels, 16 Pages
(For the Pro)
Complete Z80 Instructions.
Assembler Instructions, Commands, Operators.
Editor/Assembler Commands, and Edit Subcommands.
Flags, Conditions, & Chart.
Internal Routines.
Assembler Error Msgs.
Plus Most Items in the Basic Card

MODEL II

BASIC & ASSEMBLER: Green
10 Panels, 20 Pages
(For the Business)
Small Memory Map.
Screen Layout.
Easy Graphics.
Complete Z80 Instructions.
Series-1 Assembler Instr.
Commands, Operators, and Edit Subcommands.
Assembler Error Msgs.
Power-up Error Msgs.
Flags, Conditions, & Chart.
Wild Cards, DOS Messages.
SVC Procedure Panel.
Host Logon Panel.
Version 2.0 Lib Command Formats and System Utility Formats.
Basic Functions & Statements.
DOS File Naming Convention.
Basic Commands & Edit Subcommands.
Special Keys.
Basic Internal Codes and Reserved Words.
Basic Msgs. & Codes.
PRINT USING Examples.
Special Characters.
"DO" Utilities & BASIC Command.
Ascii Character Chart with SVC Names and Numbers.
Control Codes.

MODEL III

BASIC: Blue & Buff
6 Panels, 12 Pages
(For the Classroom)
Special Characters.
Kana Characters.
Euro-Characters.
Memory Map.
Special Keyboard Functions.
Ascii Char. Chart w/Space Compression Codes.
Control Codes.
Cassette Loading Err Msgs.
Basic Commands, Edit Subcommands, Special Chars., Basic Statements, Facts, Functions, Derived Functions, Special Operations (POKEs).
PRINT USING Examples.
Basic Msgs. & Codes.
Basic Internal Codes.
Reserved Words.
Screen Line Layout.

BASIC & ASSEMBLER: Blue
10 Panels, 20 Pages
(For the Pro)
Complete Z80 Instructions.
Assembler Instructions, Commands, Operators.
Series I Editor/Assembler Commands & Edit Subcommands.
Flags, Conditions, & Chart.
Hex/Dec Conversion Chart.
Assembler Error Msgs.
Internal CALL Routines.
Break Processing Procedure.
Plus all items in the Basic card.

COLOR

BASIC & EXTENDED: Grey + 9 Colors.
8 Panels 16 pages
(For the Artist)
All Color Graphics.
System Commands.
PRINT USING Examples.
Special Characters.
Special Keys.
Cassette Loading Err Msgs.
Basic Functions & Statements.
Playing Music, Making a Circle, and Drawing Panels.
Derived Functions.
Messages & Codes.
Musical Notes, by Octave, in Color, Including Rests and Time.
Memory Map.
Reserved Words.
Internal Codes.
A Page of Tips.
Ascii Char. Codes Chart, Including Inverse Graphics and Color Graphics.
Control Codes.
Color Group Chart.
Pmode Information Summary.
Screen Line Layout.
Extended Graphics Pmode Illustrations.

APPLE II & II PLUS

BASIC: Red & Pink
7 Panels, 14 Pages
(For the Classroom)
48K Memory Map
APPLESOFT and INTEGER BASIC.
Basic Statements.
Basic Functions.
Derived Functions.
Special Characters & Operators.
System & Utility Commands.
Pokes, Peeks, Calls.
Monitor Commands.
Key & Control Functions.
APPLESOFT Internal Codes.
APPLESOFT Reserved Words.
Integer Basic Addressing.
DOS 3.3 Command Summary.
Color Selection Chart.
Error Msgs. & Handling.
Reading Machine Language.
Hex/Dec Conversion Chart
ASCII, Print, Video, 6502, Integer and APPLESOFT Code Reference Chart, 0-255.

Basic & 6602: Red
8 Panels, 16 Pages
(For the Pro)
All features of the Basic Card, Plus:
6502 Timing.
6502 Language Simplified.
Flags & Conditions with Reference Chart.

POCKET

BASIC: Purple
5 Panels, 10 Pages
Operating Characteristics
Memory Types and Limitations.
Modes of Operation.
PRO Mode.
RUN Mode.
RESERVE Mode.
DEF Mode.
Fixed Variable Facts and References.
System Function Keys.
Math and Logic Function Keys.
Normal Character Keys.
Special Characters and Function Keys.
Basic Commands.
Cassette Interface Commands.
Reserved Words.
Math and Numeric Functions.
Derived Functions.
Basic Statements.
Error Messages and Codes.
USING Statement Examples and more. . . . !

A pocket card for your pocket computer.

Please send me:	Card	Price
( )	Copies of MODEL I BASIC & ASSEMBLER	$4.95
( )	Copies of MODEL I BASIC-ONLY	2.95
( )	Copies of MODEL II BASIC & ASSEMBLER	5.95
( )	Copies of MODEL III BASIC & ASSEMBLER	5.95
( )	Copies of MODEL III BASIC-ONLY	3.95
( )	Copies of COLOR BASIC AND EXTENDED	4.95
( )	Copies of POCKET BASIC	2.95
( )	Copies of APPLE II & II PLUS BASIC	3.95
( )	Copies of APPLE II & II PLUS BASIC & 6502	4.95

Wholesale prices available in quantities over 24.

Send Check or Money Order to:
NANOS SYSTEMS CORP.
P.O. BOX 24344
SPEEDWAY, IN 46224
(317) 244-4078

NAME: _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

Add $1.00 per card for shipping and handling. Indiana residents add 5 percent for Indiana sales tax.
Available at most B. Dalton's and other fine stores across the country.

Letters to the editor

Cameron C. Brown

I've enclosed my company check for a space ad. I look forward to seeing what type of results it gets in *Basic Computing*. I'm a subscriber myself and pour over every issue. Never before have I found a magazine with more useable information per issue than *Basic Computing*. My computer and I are very pleased.

Bob Keller
Stanton, CA

There have been so many experiences with suppliers that have been unpleasant since I bought my computer in 1978, it is especially pleasant to report a satisfying one. *Basic Computing* had a part in it.

I use VisiCalc to create a trading model for investments and the September, 1983 issue discussed a product called Liason, from Kjell Engineering, in the Exploring VisiCalc column. I ordered the product and described my special needs for formats. Two days later I got a call from Dave Kjell telling me he thought he could provide a solution for me and asked if I would want him to do so. The happy result is that I now have a very elegant program that cust fours days work down to twenty minutes.

In addition, Mr. Kjell has made a number of calls to provide some quick fixes, pick me up where I stumble, and advise me how to rectify some hardware problems. His skill, service and personal attention were of the highest caliber. Thank you for providing the information that enabled me to arrive at this happy solution.

Roger Bass
Carslbad, CA

Help! I purchased the Family Tree program from Computer Shack for my Model III, tape version. I can run the program but, when I try to load data from tape, individual person numbers no longer match recorded names. Talks with Computer Shack and the author have not given a solution. I would appreciate hearing from any readers who may have found a solution.

Walter A. Bailey
4701 W. Duval Rd.
Austin, TX 78727

Notes, etc.

Cameron C. Brown

Our New Tandy 2000

We now have a 128K, two disk drive, Tandy 2000 with a high resolution board and color monitor. We can't get Greg, our technical editor, to work on anything else. The machine is quite a beautiful piece of hardware. Going from TRSDOS, LDOS, DOSPLUS, or NEWDOS to MS-DOS is going to take me some time. The new operating system presents quite a challenge to someone who began computing with a Model I, Level I machine.

Here are some early discoveries we can pass on to you. Depressing the Print key will lock-up the keyboard if a printer is not attached. There is no error message (such as "printer not ready"), just silence. Pressing the Print key a second time will bring back keyboard control. Press any other key and you have to reset. There may be a way to error trap this, but it should be part of the DOS. The problem is serious since the Print key is right next to the backslash key that you will use quite often to change directories.

The Tandy 2000 help number in Fort Worth is (817) 390-3471. Perhaps by the time this is in print they will have had a chance to read their technical manuals. When we called in early January, they knew about as much about the machine as we did. And that is not saying much.

The interplay between MS-DOS and BASIC is curious. You can set printer width and length in BASIC, but we didn't find an MS-DOS command to do it. It appears that you can't execute system commands such as dir from within BASIC. We also have yet to find a way to return to a BASIC program after going to the MS-DOS level. I am sure that small routines could be written to do all of that and more, but they strike me as rather obvious features that should be there.

Greg has found that there are some undocumented commands in the IBM-PC BASIC 2.0 (see *Softalk for the IBM Personal Computer*, January 1984) but they are not in the Tandy 2000's BASIC. One of them is SHELL, another is ENVIRON. It just points out again that IBM-PC software which does not follow MS-DOS 2.0 documented rules will have trouble running on the 2000. Please remember, all of these notes are preliminary; there are many pages of documentation to go through.

If you have a Tandy 2000 and wish to review software please send us a letter describing your system, interests, and background. Most of the packages for the Tandy 2000 are quite extensive and we need to expand our reviewing staff to cover them.

Tips and updates

My thanks to the readers who send in these notes; you are the source of much of the information in this column.

Mr. Harold Laroff wrote to remind us to put in a remark statement or two in any program that we enter from a magazine listing that tells which issue it came from. That certainly does make referencing articles a lot easier.

In attempting to aid a reader with unidirectional printing on an Epson printer, Mr. Tim Bowman found five errors in the hex codes that were listed in the appendix of the manual for the MX Printer. The manual is entitled *MX Printer Manual with Graftrax Plus*, and is identified with the numbers P8294003-2 and Y 422991140 on the cover. The set column width command, ESC Q, is hex 51, not 5B. The set superscript/subscript mode, ESC S, is hex 53, not 5D. The reset superscript/subscript mode, ESC T, is hex 54, not 5E. The unidirectional printing mode, ESC U, is hex 55, not 5F. The double-width printing mode, ESC W, is hex 57, not 61. For readers who were having trouble with the December Exploring VisiCalc column on setting printer modes, and who own an Epson printer, this may solve your problems.

Mr. John Roland sent in the following code in response to the November, 1983 Notes, etc., item regarding the VAL command. Notice what happens when a double precision decimal numeral is put into VAL with a single precision designator.

```
10  CLEAR50: DEFSTRA:
DEFDBLN
20  INPUT A
30  N = VAL(A)
40  PRINT N
50  GOTO 20
```

Good Company Award

The user-friendly reader response card in the back of this issue asks for

your vote for the company that has provided you with excellent service and response. Be sure to get your vote in by April 6, 1984. Remember, we are looking for service and after-sale support. Many companies have an outstanding product, but we want to know which company has impressed you with their help, response, and service.

Corrections and Updates

Model 100 Calendar Program, December, 1983, page 59 has been enhanced by a number of readers. Mr. Robert Modrcin of Overland Park, Kansas changed lines 12, 17, and 18 to make today's date appear in reverse video. His new lines were:

```
12 L$="312831303130313130313031": YY=VAL(MID$(DATE$7,2))+1900: M=VAL(MID$(DATE$,1,2)): D1=VAL(MID$(DATE$,4,2)): IFM=0THENMENU
17 FORI=1TOVAL(MID$(L$,M*2-1,2)): P=I+W-1: IFP/7=INT(P/7)THENPRINT: PRINTCHR$(27)+"q";: PRINTTAB(6);
18 PRINTCHR$(27)+ ''p'';: IFI<>D1THENPRINTCHR$(27)+"q";: IFI<>10THENPRINTCHR$(32);
```

Mr. Lyndon Mitchell, the author of the original program, sent in a new listing that allows the calendar to begin with any month that you choose. His new program is shown in Listing 1.

Puzzler

Our December Puzzler asked for which integer, 5 to 80 inclusive, has the largest number of steps required to give a palindrome by adding a number to its reversal (the process is continued on the sum until a palindrome is reached). The correct answer is 79 and it palindromes in six steps. The sequence is 79, 176, 847, 1595, 7546, 14003, 44044. We had many correct solutions and the winner, selected at random from those that were correct, is Mr. Paul Hine of San Diego, CA. Our congratulations to him and we hope he enjoys his free six-month extension to his subscription.

This month we want to try something just for fun. There have been numerous programs written that will convert a number from one base to another, or from Arabic to Roman, or vice versa. I never really used Roman numerals, but I do use a simple tally system. Remember the technique of making four vertical marks, and then crossing them out diagonally when you reach five? How about a program that will take the input of a positive decimal integer and print it out on paper in a tally form? We won't be picky about the diagonal line, just devise a method that makes the groups of five distinct. The capital letter I would be acceptable for the vertical bar if your printer can't do it. Send your code and a sample printout to March Puzzler, c/o *Basic Computing*, 3838 So. Warner, Tacoma, WA 98409. The author of the winning solution will receive a free six-month subscription extension.

Listing 1

```
100 'CALEND.BA BY LYNDON B. MITCHELL 831203130000
110 L$="312831303130313130313031"
120 YY=VAL(MID$(DATE$,7,2))+1900
121 INPUT"YEAR";Y
122 IF YY<1900 THEN YY=YY+1900
123 IF YY<1900 OR YY>2099 THEN 121
130 M=VAL(MID$(DATE$,1,2))
134 INPUT"MONTH";M
135 IF M<1 OR M>12 THEN 134
140 IF M=0 THEN MENU
150 CLS
160 Y=YY-1901
170 L=INT((YY/4-INT(YY/4))*4+.5)
180 D=Y*365+INT(Y/4)+1
190 IF M=1 THEN 240
200 FOR I=1 TO M-1
210 D=D+VAL(MID$(L$,I*2-1,2))
220 NEXT I
230 IF L<>0 THEN 250
240 IF M>2 THEN D=D+1
250 W=INT((D/7-INT(D/7))*7+.5)+1
260 PRINT TAB(16);MID$("JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC",M*3-2,3);YY
270 PRINT TAB(6);"SUN MON TUE WED THU FRI SAT"
280 PRINT TAB(6+W*4);
290 FOR I=1 TO VAL(MID$(L$,M*2-1,2))
300 P=I+W-1
310 IF P/7=INT(P/7) THEN PRINT:PRINT TAB(6);
320 IF I<10 THEN PRINT CHR$(32);
330 PRINT I;
340 NEXT I
350 IF L<>0 THEN 370
360 IF M=2 THEN PRINT " 29";
370 K$=INKEY$
380 IF K$="" THEN 370
390 IF ASC(K$)=77 THEN MENU
400 IF ASC(K$)<30 OR ASC(K$)>31 THEN 370
410 IF ASC(K$)=30 THEN M=M+1:IF M>12 THEN M=1:YY=YY+1
420 IF ASC(K$)=31 THEN M=M-1:IF M<1 THEN M=12:YY=YY-1
430 GOTO 150
```

Taking a program to court

How to win a Small Claims Court case for a defective program

For all readers

Kelly Hauptmann

Everything you are about to read is true. I have purposely disguised the names and locations of the principal actors in this little skit, because I have no intention of being sued for libel. *I am not a lawyer!* I am not intending to give you legal advice. I am only relating a true-life story as it happened and as I, a layman, perceived it. *I am not giving legal advice.* (The above emphasis on the advice of my lawyer.)

My name is Kelly Hauptmann. I live in Smalltown, in the state of Calizona. I am the president and majority stockholder of the Veeblefelster Mfg. Co., Inc. Veeblefelster is a company that makes widgets, ferpils and freeps for sale to people who put them in their geewhizzes and drive them around. I (the corporate "me") own a TRS-80 Model X, on which I have my inventory, my sales data, and my invoicing. Some time ago, I purchased a program from Snerdly Whippo of Hugeville, state of Calizona, to perform the inventory, sales and invoicing functions.

Whippo advertised his program in a large number of national computer magazines as being written for the Model X, and the ad copy said that the program was intended to do inventory, sales data, and invoicing (here I quote, because it later became important), "without error." Now, you and I know that there has never been (and probably never will be) a program written without its share of bugs. However, most reasonable people expect the bugs to be hard to access — that one-in-a-million-shot where the quantity has to be exactly 589, the color red, with the optional pinstripes. Not so. Whippo's program (called, for lack of a better name, Comprehensive Business Program, or CBP for short) refused to add invoices correctly. It lost whole sections of inventory randomly, without apparent rhyme or reason. Sales would be in error by several thousand dollars a month (plus or minus, depending on the phase of the moon, amount of rainfall, or who was leading the National League west). We sent CPB back for "fixes" half a dozen times. Each time, the problem would come back fixed, only to find that the "fix" had generated more problems. After we had invested five thou$and dollars in time, we decided that the program would continue to be a

problem child for us, that the author had led us down the primrose path, and that we wanted our money back. We sent him back the program disk, documentation and a demand for a refund. The total time from first receipt of the program until we sent it back was the span of six months.

Of course, Whippo refused to send us a refund. After much deliberation and many telephone calls (none of us really wants to be nasty to our fellow human beings), we decided that we had been "had," and that the easiest route for us to take to recover our original outlay for the program was to sue Whippo in Small Claims Court. Note — this court goes by many names in the several states: Justice Court, Small Claims, People's Court, and the rules are quite similar. A relatively small amount of money damages may be claimed (Calizonia $1500); the damages must be money, not property; you may *not* have a lawyer represent you; if the person sued (defendant) loses, they may appeal to a higher court; the rules of evidence (indeed the whole proceedings) are very informal as opposed to a full-blown judge-and-lawyer-type trial.

We had to sue Whippo in the Hugeville courts (his hometown) because of the nature of contract. As it was explained to me, the relationship between Whippo and my company was in fact a contract, which consisted of two essential parts: offer and acceptance. You see, the offer was made when I put the order blank and check into the mail in Smalltown, but the contract was not completed until Whippo accepted my check and sent me the program. Since the contract was thus concluded in Hugeville, the Small Claims suit must be brought in the Hugeville courts. Please note that this was inconvenient as the deuce on me, as I had to travel (twice, as will be shown later) from Smalltown to Hugeville, some 1000 miles round-trip, for the trial.

The trial came to pass one bright spring morning in May. I had been advised to wear neither the best suit in my closet (the judge would assume that I was a rich merchant beating up on a poor defenseless tradesman) nor crummy jeans and a T-shirt (the judge would think I was a bum trying to take advantage of an honest merchant), but rather a nice, simple outfit. I chose a white turtleneck (purity and chasteness) and a grey sports coat and pants (middle-class worker). Laugh if you will — what the judge thinks of you is cotton pickin' near as important as what you have to say.

All these preliminaries aside, the trial opened with a pledge of allegiance, and we were "in session, the Honorable Tinkerputter Q. Doofus presiding." As the plaintiff, I told my story first. Doofus seemed impressed that I had copies of all my correspondence with Whippo, that I had the original magazine ad that I saw with the claim of error-free record keeping, and that I did not try to "snow" him with computer jargon. Believe me, the judge was much happier when I argued that this was a simple breach of contract as opposed to arguing that it was some kind of newfangled "computer case." Judges have had years of experience on contract law. They are happy with contracts. They view computers as some sort of alien force that they have no control over. They are so happy to hear that this "computer case" is nothing more than simple contract breach that they will listen to any argument that mentions "contract" with open ears, while bits, bytes, and RAM sink gently to the floor and are swept out the back door.

At any rate, I told my story, Whippo told his story, and the judge asked us both a few questions, mostly in the field of warranty. Warranty, as my feeble mind understands it, is the promise by the seller that his product will do what he says it will, and what he will do about it if it doesn't. It seems that if the warranty is not an item which is part of the original bargain (that is, if neither I nor Whippo made any agreements as to the terms of the warranty before the contract was concluded) then the terms of any warranty are decided by the statutes of the state and the circumstances that the famous hypothetical "reasonable man" would assume apply. Since neither of us mentioned word one about warranty prior to the delivery of the program, the judge presumed that what Whippo stated in his ads was what he would

deliver.

The judge told us that he would deliver his judgment by mail, and that we were both dismissed from the courtroom. Both Whippo and I departed the courtroom. Outside, we shook hands and said, "May the best person win," even though both of us believed that he had won. Honestly, Whippo had some very good arguments — his company had used the program (admittedly on a Model Y) for some years; his secretary testified that the program had run on the Model Y without any problems for some time and he, himself, appeared as an honest programmer that had made a number of honest mistakes.

Three weeks later, the court rendered a verdict by mail for the plaintiffs (us) for the full amount of the program (some $700) plus our court costs of some $30. About two weeks later, we were somewhat surprised to be served with papers showing that Whippo had appealed the Small Claims judgment to the Hugeville Superior Court. This, of course, was his right as a defendant who lost, but it meant a whole new trial, with far-reaching ramifications.

In the state of Calizonia (typical of most states), an appeal from a Small Claims decision by the defendant is heard by the Superior Court as a 'trial de novo.' A literal translation of the Latin means that we do the whole thing all over again, right from scratch. The bad news is that a corporate officer (i.e., me) cannot represent Veeblefelster, Inc. (a corporation) in Superior Court, although I could represent them in Small Claims court. Don't ask me the logic. There is none. We are talking law here, not logic. The bottom line is that I had to impose upon an old friendship with a "licensed attorney" to represent the corporation in a piddling little matter like this $700 suit.

The Superior Court of Hugeville County spent exactly one minute and thirty-seven seconds re-examining the findings of the Small Claims Court. The verdict was the same — Whippo owed my company the $700 for the program, plus $20 in Small Claims fees, plus $15 for my lawyer in the Superior Court appeal (more about this later). Since Sam (my lawyer) had to travel 150 miles, put in a couple of hours in preparation for the case, and no further testimony or evidence was introduced, how the judge figured Sam only earned $15 for his services is a mystery (mostly to Sam) to this day.

A week later, we received a check from Whippo in the amount of $735; $700 for the program, $20 for court costs, and $15 for my lawyer. Sam laughed the lawyer's award off and said that some day he will tag us for his time. Today he is pleased that justice has triumphed.

This whole experience has taught me a bunch of lessons that you might do well to heed:

1. Keep copies (on magnetic media if possible) of all correspondence between yourself and the programmer.

2. Keep copies of any magazine ads on the program. Keep copies of any documentation received with the program. Keep copies of phone bills and post office receipts showing contact with the programmer. Keep copies of your laundry list if you think it will have any bearing whatsoever on the case. The judge just loves receipts, ads, and records.

3. Believe that the programmer has one primary interest at heart, no matter how sympathetic (s)he is to your problems. The primary interest, I guarantee, is not your problems.

4. Do not bug or sue the programmer for piddling errors. If the disk file is supposed to handle 100,293 files and it has errors at 100,291, give the programmer a shot at correcting the small error.

5. If you are a corporation, the courts have strange and screwy rules about who and where the owners of a small corporation can sue an individual. Much better that you sue as an individual shareholder of the corporation (even though you may be the majority or sole shareholder) than the corporation sue as itself. As a shareholder, you can sue right up to the Supreme Court of the United States without the services of a lawyer (in pro per); the "corporation" must retain (expensive) licensed legal counsel at the appellate level.

6. Remember that the basic legal premise is that of equity — did he give me an equal measure of what I gave him? Not, did you get your money's worth? It is your job to reasonably ascertain that what he has to sell is what you need. Not, did the program perform your task? It is your job to find out to the best of your ability before purchasing the program that it will do your work properly. It is the task of the programmer to adequately describe and delineate his program, and it is the task of the user to take this description and delineation and decide whether or not the program will perform his job adequately.

7. If you really believe you have a case, go for it. However, remember that the defendant (programmer) has a chance to cross-sue for fraudulent or whimsical damages, so believe in your own stuff before proceeding.

8. If you believe you are entitled to $700, and the limit on small claims in your state is $1500, go for the maximum. Put your wasted time, your travel expenses to and from the court, your wasted computer time, telephone calls, paper clips, everything remotely affected by the defective program, into your request for judgment. Let the judge sort out how much you are really entitled to. That's his job; that's why he gets paid to sort out who owes whom what and why. In our case, the judge would have been more than willing to give us a few hundred bucks in travel and lawyer expenses, but since we didn't ask for them in the original suit, he couldn't grant anything additional.

In closing, remember Ben Franklin's admonition that the person who represents himself has a fool for an attorney, and get the legal advice you need. In our case, we might have tagged that rascal Whippo for several thousand dollars more, had we engaged the services of an attorney at the outset, for consequential damages and all that rot. However, most of those thousands would have gone to the attorney (given the crapshoot that we won), and had we lost, the attorney's bill would have been just as huge. It may pay you to search out an attorney who will serve for a percentage of the win, and a basic retainer in the case of loss.

SOFTWARE PRICE WAR

$19.95

These are complete software systems supplied to you on a **TRSDOS 1.3** system diskette
All systems fully menu driven — user friendly just insert diskettes, reset and go. Systems are supplied with full documentation (although most users report that they don't read it).
Easy backup from main menu.

This is a sample of my mail

"Your PAYROLL program is great — I use it for all my accounts — it's quite a time saver." Hampton, NJ

"I received your PAYROLL program yesterday and was impressed by it. Hence I am enclosing a Cashier's check for..." Jacksonville Beach, FL

"A friend showed off his possession of your check register system to me and I want a copy for my personal use." Bloomington, IL

"We have received the first two programs and have found them to be quite sophisticated and are anxious to try another. Please accept my order for..." Durango, CO

All systems are written for Model III or Model IV in Model III Mode.

LYNN'S PAYROLL SYSTEM

Complete record keeping & payroll calculation. To include your state's unemployment tax & forms, detailed 941 form, prints paychecks & W-2's. Accounting write-up version available.

LYNN'S ACCOUNTS RECEIVABLE SYSTEM

Single entry — open invoice system. Prints invoice and statements. Built in aging report.

LYNN'S CHECK REGISTER SYSTEM

Prints checks. Register for any month, reconciliation statement.
1,000 checks per month— 220 expense accounts.

LYNN'S TRSDOS UTILITY PROGRAMS

For TRSDOS 1.3 includes short directory (file names only), cursor directory (allows cursor access to CMD, BASIC, and "DO" Files), and a 2K Print Spooler. Menu installs utilities on existing TRSDOS Diskettes.

LYNN'S E-Z MAIL SYSTEM

Sort by name, zip code, state. 2,500 names. E-Z edit.

AND NOW THE BEST PART

ANY SYSTEM ONLY $19.95 EACH.

Plus $3.00 shipping per order. (add $1.65 for C.O.D.) Ill. residents add 6¼% sales tax.
(The **TRSDOS** alone is worth **$14.95**)

master charge THE INTERBANK CARD

#8

Joe Lynn
Computer Services
23501 W. Gagne Lane
Plainfield, Illinois 60544
815/436-4477

BANKAMERICARD VISA

The compatibility game

Developing software that is machine independent

For all readers

How many times have you bought a piece of software, supposedly the "latest and greatest," only to find that it won't run on your combination of machine, DOS, density, printer or whatever? If you've lived through this frustration, this is an article for you. If you enjoy programming, hopefully the ideas presented below will keep your programs out of trouble. If you primarily buy other people's software, this should give you some ideas of what to look for.

First, we need a very quick review of microcomputer history. When micros first came on the scene in the early 1970s, it was pretty much a "do-it-yourself" kind of industry in much the same way ham radio was in its early days. There was precious little hardware to be had (old Teletype machines and paper-tape readers were the vogue), and even less pre-packaged software. Operating systems existed only in the minicomputer and mainframe environments. If you wanted to write a program it was up to you to also write your own programs to run the actual hardware. Since no systems software standards existed, everybody did as he or she pleased. That resulted in a minimal amount of software portability between systems.

The first truly popular microcomputer operating system was CP/M, written by Dr. Gary Kildall. For the first time it was possible to write programs to be "machine independent." In other words, you had the option of writing a program on one machine and then running it on another. In a real sense, CP/M was responsible for the software explosion we've experienced ever since. In fact, even though substantially more powerful operating systems have been introduced, CP/M has been the most popular 8-bit DOS to date. No doubt this is because of the huge amount of software designed with CP/M in mind.

The System Concept

The key to CP/M's success in making programs portable was its implementation of what we'll call the "logical device." Let's say you wanted to write a program which printed reports. Without an operating system, it would be up to you to first check to see if the printer hardware was ready and, if so, send the data to it. With an operating system in place, however, you can send the data to a "logical" printer instead of the actual hardware. This logical printer is actually a subroutine somewhere in the operating system that you call from your program. This subroutine takes the data to be printed and sends it to the physical printer. It does this by making use of a program built into the DOS called a device driver. The device driver is the one program that changes when you implement the DOS on a new machine. The driver takes a command from the DOS such as "Print The Data" and customizes it as necessary for a particular piece of hardware.

> There is little reason for incompatibilities of this sort to continue.

Now, let's say we write different versions of the DOS so it can run on various machines. As long as we keep the location of the logical printer the same in each version and only change the device driver programs, a program written on one machine will run just fine on another. The only exception to this is if your program depends on some special feature of the printer. For example, if you send the codes for expanded print on an Epson MX-80, you will get unpredictable results using any other printer even if you talk to it via the logical printer device.

The key idea here is that of standardization. As long as you use the operating system's set of standard I/O commands, rather than trying to directly go to the various pieces of hardware, you'll enjoy a high level of portability with your software. This idea of standard I/O applies to all peripherals hooked up to the system. In any TRSDOS-compatible system, for example, there is always a device call within the DOS which allows you to "talk" to the printer, video, keyboard, and disks without actually dealing with the hardware.

This is the single area which causes the greatest grief in trying to get a particular piece of software to run on a TRS-80. For some reason, there are many programmers who choose to ignore the documented

ROM and DOS calls for their I/O operations. They have programs going directly to the hardware for I/O and invariably such programs fail to work when a new DOS, printer, or whatever is added to the system. In the early days of the TRS-80 (does anybody remember the 4K Level I . . .), Tandy was reluctant to tell us their "inner secrets" and entry points to the first versions of TRSDOS. So, older programs can be forgiven for not making full use of the operating system. Lately, however, our friends in Fort Worth have been a lot more responsive with this kind of technical documentation. There is little reason for incompatibilities of this sort to continue.

Even with documented DOS I/O calls, some compatibility problems do exist. The most notable of these is doing Disk I/O under different operating systems. Though there are disk I/O entry points common to all TRS-80 DOS products, there are slight differences in how they actually work. Here, you have to make a choice based on what your favorite DOS is. If you plan to commercially market a program, the choice is simple. You must be compatible with either TRSDOS, LDOS, or CP/M. These are the only operating systems sanctioned by Tandy. Remember that the majority of end users could care less about what the DOS is or does. They are likely to buy whatever their Radio Shack store has for sale.

Here are some specific ideas on how to write software for maximum compatibility. This material appeared in a slightly different form in an article I wrote for the *LDOS Quarterly Magazine*, and is used with the kind permission of Logical Systems, Inc. For this reason, I've used the LDOS nomenclature where necessary. This material is intended for the assembly language programmer, but those of you who work in BASIC and like to POKE and PEEK your way around can benefit too.

As we just mentioned, use system I/O calls wherever possible. This should be 90+ percent of the time. About the only major software product which legitimately needs to do physical I/O itself is a word processor. There is no TRSDOS-compatible call which will move a whole screen of data to or from the video, so the word processor must update the screen itself. Interestingly, Radio Shack is one of the greatest offenders in this regard. Products like Scripsit and SuperScripsit blindly ignore the built-in device control blocks for the printer and keyboard as well as doing the mandatory screen I/O themselves. (A Device Control Block, or DCB, is just a small part of memory which stores the address of the device driver and other related information about the device. To access a particular logical device, you load the DE register with that device's DCB address, and then call a ROM address to either send or receive a byte of data to/from that device. You can also send control information to the device this way. This technique of "Byte I/O" is discussed thoroughly in the LDOS technical manual.) In all fairness, I know of *no* TRS-80-compatible word processor which does use these system calls. This can be a real *pain*! In LDOS, for example, not honoring the DCBs for the keyboard results in a loss of all advanced DOS keyboard functions within the word processor (like "soft-keys" and MiniDOS). Not honoring the printer DCB causes the spooler not to work. The list is endless.

If you *must* do your own physical I/O, do it the right way. This is most important when you work with devices already defined in the system like the keyboard, printer, and so on. Make your program "transparent" to the DOS. When the program is first started, it should store the current information held in the DCB of the device you are going to work with. That way, you can restore this information before your program returns control to the DOS. The system will never know you were there. Once you've stored this information, load the DCB with the location of your own device driver. That way, you can still use standard system I/O calls in conjunction with your own device handling procedures.

When you work with physical I/O yourself, be absolutely sure your procedures are right. One of my favorite examples shows how many programs try to detect printer status on a Model I (Figure 1). This is absolutely *wrong*! The reason is that printer status on a Model I is held in the upper nibble only. The lower nibble has never been officially defined and, in fact, varies on some interfaces like the LOBO LX-80. The corrected version is shown in Figure 2.

This brings us to point four. *Never* use undocumented features of either software or hardware in your programs. For example, I recently read a letter from someone who was infuriated that some of the new DOS products will not reboot if you do a JP 0000 at the end of a program. There's a good reason for this. That exit was never officially defined by Tandy. The correct way to exit a program on a Model I or III is to JP 402DH.

At the beginning of your program, you should preserve the contents of every register you intend to use. Likewise, you should restore them just before you exit your program. The easiest way to do this is to PUSH all the register contents onto the stack when you start a program, and to POP them back at the end. In fact, if you have an assembler like EDAS IV, MACRO-80, or EDT-ASM+, you can write macros to do this for you with one-line commands. Preservation of registers is part of the concept of making your programs transparent to the DOS. In many cases, failing to do so will cause the DOS to blow up when your program returns control to the system.

Never use the Z-80 alternate registers (AF', HL', etc.) for your own code. These are reserved for use by the operating system. For example, some systems use these registers during interrupt handling. If you use them, the system will invariably fail to perform properly. The Z-80 stack is a more appropriate place to store data which you need to access rapidly.

Finally, honor the high memory pointer in the DOS and make all your high memory programs self-relocating. A TRSDOS-compatible DOS maintains a pointer called HIGH$ which tells the system the highest unused memory location. HIGH$ is at X'4049' in a Model I system and X'4411' in a Model III system. A properly written program will never attempt to use memory higher than the current value stored in HIGH$. This is why memory above the current value in HIGH$ is referred to as Protected. Programs such as special device drivers, I/O filters, and machine language sorts are often placed in protected memory to keep them from being overwritten by the main program that is currently executing. Here's where we run into the second part of the problem. Suppose you have two high-memory programs, say a printer driver and a machine language sort for a data base manager. If they both are designed to load at X'F0000', you have an obvious problem! Only one program

can occupy a given part of memory at one time. The solution is to make all your high-memory programs self-relocating. Here's the basic procedure for doing this:

1) ORG your program to start at some low memory location (usually X'5200').

2) Have your program examine the current value of HIGH$ and decrement it by the length of the program to be placed in high memory. This will protect the program once you've moved it to its final location.

3) Resolve all absolute address references inside the program to reflect the fact that it is going to be moved to a new starting address. This would include any CALL, JP, or LD instruction which refers to a location within your program. For example, let's say you ORG the program at X'5200' and the first instruction is JP 5300H. Suppose also that after step 2 above, your program determines that it is to be relocated starting at X'F100'. You would then have to change the first instruction to be a JP F200H.

4) Finally, move your program to the new high, protected memory. This is easily done with the Z-80 LDIR instruction.

Keep in mind that all four of these steps are done each time you load the program (i.e., at runtime). If you follow this kind of procedure, the current value of HIGH$ becomes unimportant. Your program will always relocate itself to memory just below the present value of HIGH$ and then decrement HIGH$ to protect itself.

Figure 1

```
LOOP    LD      A,(37E8H)       ; GET PR STATUS
        CP      3FH             ; IS IT READY?
        JR      NZ,LOOP         ; IF NOT, TRY AGAIN
```

Figure 2

```
LOOP    LD      A,(37E8H)       ; GET PR STATUS
        AND     0F0H            ; MASK LOW NIBBLE
        CP      30H             ; IS IT READY?
        JR      NZ,LOOP         ; IF NOT, TRY AGAIN
```

This is a common cause of incompatibility in various systems. The majority of applications software I've seen ignores HIGH$ altogether. Programs which are self-relocating are even rarer. If you feel writing self-relocating code is beyond your abilities, there is another option. XYZT Computer Dimensions in New York sells a package called RELO, which will take your programs and make them self-relocating.

In summary, let me give you an example of how important all these ideas are. Lobo Systems in California introduced a Model III work-alike computer called the MAX-80. What is interesting is that the actual hardware resembles a Model I, but the operating system calls and entry points are Model III-compatible. If, for example, a Model III program tries to access the printer directly through the I/O port on this machine, the program will fail. However, if the programmer does things right and accesses the printer via the DCB, everything works just fine.

This list of ideas is not exhaustive. There are many other techniques which can be used to make software more portable. However, keeping these ideas in mind as you design new software will eliminate the majority of compatibility problems.

Apple-80

How to convert Apple programs to work on a TRS-80

For all readers

David Lewis

I believe that the TRS-80s are superior to the Apple II or Apple II+, but those machines have sold better than Tandy's best. Why? Well, it has been said that the software sells the hardware, and there are more programs available for the Apple than for any other microcomputer.

You may have thought that all that software was out of reach, but both the Apple and the TRS-80 run versions of Microsoft BASIC. There are differences between the two dialects, but many can be overcome. So, then, here is a comprehensive guide to converting "Applesoft" programs to run on your Model I, III, or 4, or on your LNW, LOBO, or PMC. (Some of the guide is also appropriate for the Model 100, for the Models II/12/16, for the Color Computer, and MC-10.) You might even learn more about your own system while doing it.

Input/Output Commands

The Apple can send or receive data through the keyboard, through expansion slots similar to those on the Model 12, or through input and output ports.

The Keyboard

Even such an innocuous command as INPUT has its differences. The Apple does not automatically insert a question mark when text to be printed follows the INPUT. When translating a line such as:

```
10 INPUT "WHAT IS YOUR NAME? ";NM$
```

be sure to remove the question mark, or you'll wind up with two.

The Apple also has a command similar to INKEY$. The GET statement, however, waits for a single key to be pressed. If you see a line such as 10 GET A$, convert it with something like:

```
10 A$=INKEY$: IF A$="" THEN 10
```

This TRS-80 line will continually loop until a key is pressed. (GET and INPUT are also used with text files; they are essentially equivalent to INPUT#. More on this later.)

The Apple makes up for the deficiency in the GET command by providing a keyboard latch, which does respond instantly. PEEK (-16384) returns a value of 128 plus the ASCII value of the character pressed. If the value is less than 128, no key has been pressed; be sure to delete the line that checks for this. Because of its responsiveness, PEEK (-16384) is frequently used in keyboard-controlled games. Another use is with the WAIT command. WAIT X,Y,Z waits until the contents of location X, when XORed with Z and ANDed with Y, gives non-zero results. In practice, WAIT -16384, 128 pauses until a key is pressed. It can be simulated by:

```
10 IF INKEY$="" THEN 10.
```

PEEK (-16384) can also be translated by using INKEY$:

```
10 X= ASC (INKEY$) + 128.
```

Whenever address -16384 is used on the Apple, the keyboard strobe must be cleared. You can ignore the POKE -16368,0 that accomplishes this.

Paddles and Joysticks

The Apple has special joystick and paddle ports. (Oh, to have bought a business machine!) PDL(x) returns the current value (from 0 to 255) of the game control numbered x. Two separate readings, usually when x=0 and x=1, are needed for a joystick. This function can be simulated by using the input ports on the TRS-80. The manufacturers of your joystick may say that INP(236), for example, is equal to PDL(0).

In addition, the Apple can determine if a fire button is being pressed. If PEEK (X-16287) is larger than 127, then the button on control x is being pressed. This, too, is simulated by using INP(port) on the TRS-80.

Expansion Slots

The Apple has 8 slots, numbered 0 to 7. IN#X causes all input to be received from slot X, and PR#X transfers all output to slot X. Traditionally, the printer is in slot 1. If you see a PR#1, it means that whatever is printed on the screen will also go to the printer. This is one major advantage of the Apple over the TRS-80. We have to

make do with LPRINT. The easiest thing to do, though by no means the most efficient, is to duplicate a set of statements, using PRINT once so that output is to the screen, and LPRINT so that the printer receives the data. PR#0 turns the printer off.

If you ever see an IN#6 or PR#6 in a program, it means that the author was being terribly sneaky. Slot 6 contains the disk drive, so a PR#6 reboots the system. You needn't translate it, but the closest thing would be a CMD"S", which returns to TRSDOS on my Model III.

Read, Data, Restore

The READ statement, which allows access to data within the program, can be converted directly. The DATA is also the same, and RESTORE, which moves the pointer to the first line of data, also causes no trouble.

Cassette

Old Apples can use LOAD and SAVE to access programs on cassette tape. These are just like the CLOAD and CSAVE you are familiar with. However, you will probably never come across them. You'll also probably never see STORE and RECALL, which save and load an array from cassette tape. Apples are sold only with 48K and disk drives.

Sound

The last major form of output is the making of sound. The Apple has a built-in speaker that many programmers take demoniacal joy in clicking. The statement X=PEEK (-16336) generates a frequency of about 72 Hz. Another method that you will come across is to use CHR$(7) or CTRL-G, which causes a longer tone and which can be embedded within quotes. You might look at something like:

```
10 PRINT "YOU WIN!"
20 FOR G=1 TO 10
30 PRINT ""
40 NEXT
```

and not be able to figure out what's happening in line 30. Model I users might be more familiar with embedded control codes. There is a control-g within the quotes.

A machine language program, typically located 768 decimal, can be used to produce a wider range of sounds. You might see something like:

```
10 POKE 768, DURA
20 POKE 769, FREQ
30 CALL 770
```

which pokes the duration and frequency into memory and then calls upon the sound driver. *Don't* translate a routine that is poked into memory byte-by-byte. The Apple uses a 6502 chip which is terminally incompatible with the Z80.

The TRS-80 has no built-in speaker, but you can get sound by hooking the cassette cable into the AUX line of your cassette player. OUT 255,2 then produces a sound, of sorts. A machine language program might also be used. There's a decent one in a book Radio Shack sells called *TRS-80 Graphics*, by Kater and Thomas.

Flow of Control

The GOTO, GOSUB, RETURN, ON X GOTO, and ON X GOSUB commands can be translated without any changes. So can NEW, END, STOP, and RUN. The range of linenumbers and the length of the lines on the Apple are smaller, so no problem with the size of the program exists.

There is one command that cannot be translated. It is POP, which "pops" a return address off the stack. It is used to exit from a GOSUB routine. Try to avoid getting yourself into a position in which you must leave an address on the stack. If you do it very often, you could overwrite your variables. You might GOTO the routine and jump back; you might also see if you can't put the subroutine into a large IF-THEN-ELSE statement. If you absolutely can't change a few lines, then don't lose any sleep.

One reason that the Apple has the POP statement is because it has no ELSE, and so major decision-making subroutines can't be coded into one line but must instead form a subroutine. You can directly translate any IF-THEN statement, however. The FOR-NEXT functions as it does on the TRS-80.

A slight difference between the two versions of BASIC is in error-handling. The Apple blankets all mistakes that follow with something like:

```
10 ONERR GOTO 100.
```

The TRS-80 uses:

```
10 ON ERROR GOTO 100.
```

You will frequently see a POKE 216,0. This clears the error flag. PEEK(222) gives the number of the error. On the Apple, it is used to determine how to fix the error. It should be translated with ERR and ERL. The Apple also has only a RESUME statement. It branches back to the beginning of the statement in which the error occurred. You can translate this directly or, with minor restructuring of the program, use the more powerful RESUME NEXT or RESUME XXX.

Graphics

This is the one area of programming in which the Apple is better than the TRS-80.

Do not try to translate high-resolution graphics unless you have an LNW or Color Computer, or own a high-resolution graphics board. You will probably be able to convert a program that makes only passing use of high-resolution graphics by ignoring HCOLOR=, which sets the color to be used, HGR or HGR2, which turn the screen into graphics format, and HPLOT X,Y (to M,N), which plots a point or draws a line. Also ignore any POKEs in the range -16297 to 16304 or from -1953 to -3106, where switches for the high-resolution pages are located. The actual memory used for high-resolution graphics is 2000H to 5FFFH.

Low-resolution graphics can more easily be translated. In fact, TRS-80 graphics can access nearly four times as many blocks as Apple low-resolution can; also, we have the added advantage of being able to mix text with graphics blocks. The first thing to do is to ignore the GR and COLOR= statement, which set the mode and the color.

The Apple screen is 40 by 40, and the points are numbered 0 to 39 in both directions. The horizontal point is mentioned first. PLOT X,Y on the Apple is directly translatable into SET (X,Y). To erase a point on the

Apple, the Color is set to the background color (black is 0) and the point is replotted. If you see that this is done, substitute RESET (X,Y). SCRN(X,Y) returns the color of a point x,y. We can use POINT (X,Y) to find out if the point is lit. Both BASICs allow something like:
10 IF SCRN(X,Y) THEN PRINT "POINT X,Y IS LIT.",
in which the color is determined only to exist.

Drawing lines is somewhat more difficult. A lot of speed must be sacrificed in order to achieve the same thing on the TRS-80. The Apple allows vertical and horizontal lines to be drawn. The statement VLIN 10,20 AT 30 draws a vertical line from 30,10 to 30,20, and HLIN 10,20 AT 30 draws a horizontal line from 10,30 to 20,30. Here's how to translate them. Assume the points are in the form xLIN A,B AT C:

```
10 'VERTICAL LINE
20 FOR D= A TO B
30 SET (C,D)
40 NEXT

10 'HORIZONTAL LINE
20 FOR D=A TO B
30 SET (D,C)
40 NEXT
```

Shape Tables

Unless you have GRBASIC, shape tables are hard to duplicate on the TRS-80. Basically, the Apple allows you to create a shape by giving it directions coded into numbers that will allow it to quickly redraw a shape. Instead of plotting each point every time you want to make the shape, you can just say DRAW 1 AT X,Y. Now, the original shape (a spaceship, for example) can be packed into a string on the TRS-80 and manipulated that way. However, it is very hard indeed to ROTate it or change its SCALE. Delete all references to ROT, SCALE, DRAW, and XDRAW.

SHLOAD loads a shape table definition from tape. More common is something like:
10 PRINT CHR$(4);"BLOAD CUTESHAPE"
which probably loads a binary shape table from disk into memory. If it is followed by POKEs into addresses 232 and 233, which save the address for the table, erase the whole thing.

Screen Formatting

The Apple's screen is divided into 40 columns and 24 rows. It is not memory-mapped. Because the screen doubles as the area for graphics, TEXT is used to reset it. You may omit it in your translation.

The HOME statement is equivalent to CLS. You might also see CALL -936, a leftover from the days of the Apple I and Apple II.

The REM statement can be translated directly. Don't entirely delete these remarks; you should leave an apostrophe or the word REM at the linenumber in case it is the destination of a GOTO or GOSUB. The PRINT statement also can be translated directly. The TAB(x), associated with it, is the same.

Model II users will be familiar with SPC(x). On the Apple, it is used in conjunction with PRINT to leave a blank block of x spaces. Use STRING$(x,32) to achieve the same effect.

POS(0) returns the cursor offset from the left margin in the range 0 to 39. On the TRS-80, the cursor address is stored in addresses 16416 and 16417. To find the offset from the margin in the range 0 to 63, when X begins as 0 to 255:

```
10 X= PEEK( 16416)
20 Y= X- INT(X/64) * 64
```

This is the same as X MOD 64.

The Apple uses HTAB and VTAB to PRINT at a specified point on the screen. The argument of HTAB, which understandably stands for Horizontal TAB, is 1 to 40. VTAB has a range of 1 to 24. They can be converted to PRINT AT (or PRINT @) by scaling them to a 64 by 16 screen, though it is better to remove the blank lines. Given HTAB H and VTAB V:

```
10 X= INT (H * 64/40 + V* 16/24)
```

and PRINT @ X to get the desired location.

There are several other commands associated with PRINT. Unless you have Model 4 BASIC, with its reverse video, ignore INVERSE and FLASH. They cause what is printed to be in inverse video or flashing in time with the cursor. NORMAL, which resets the mode, can also be ignored. SPEED=x sets a printing speed. Ignore it, but consider putting a delay loop (such as 10 FOR G= 1 TO X: NEXT) at the end of each block of printed lines.

Variables

Applesoft integers are in the range -32767 to 32767. You can translate them exactly as they are — followed by a percent sign. The real numbers are in the range (positive or negative) 1E38 to 1E-38; use double-precision if necessary. The Apple uses no type-declaration tags other than % for integer and $ for string, so you can use DEFDBL A-Z. This is needed because the Apple assumes variables are real and the TRS-80 assumes that they are single-precision, so the range may not be large enough for all applications.

LET need not be translated. CLEAR is used on both the Apple and the TRS-80 to clear all variables. The variables on both have only two significant characters, so no change is needed. However, you will have to make sure that the variables you are converting do not contain any TRS-80 reserved words, such as CLS or MKI.

String lengths on the Apple do not exceed those on the TRS-80, so you may convert them directly. However, you must be sure to clear string space at the beginning of the program with something like CLEAR 500.

All string operators, including >, =, <, and + are directly translatable. There is one thing you must be extremely careful of: numeric variables on the Apple do not have leading spaces. If X$=STR$(X) on the TRS-80, you must peel off the leading space: X$=RIGHT$(X$, LEN(X$)-1).

The Cast of Characters

The Apple has no lowercase letters. Instead, the ASCII character set is duplicated. The ESCape key is CHR$(27); it is frequently used for printer commands and may be translated directly. The keyboard contains only the left and right arrows; they are given ASCII values of 8 and 9 and should cause no problem. The commands X=ASC(X$) and X$=CHR$(X) may be

translated directly.

Operators

Relational Operators

These are operators such as =, >, <, and <>. They may be translated directly. The AND on the Apple is not a bit-compare but can be used only for statements such as: 10 IF X=1 AND Y=2 THEN PRINT "X=1 AND Y=2" You can translate this AND and the OR directly.

Now, there's another tricky thing about the Apple. Zero is FALSE, as it is on the TRS-80, but TRUE is 1, not -1. If actual numeric values are used, be sure to change the sign. Also, change something like Y=(X=10) to Y=-(X=10). NOT on the Apple switches between 1 and 0; NOT 0 is 1, for example. This may also be translated, but be wary of Boolean relationships. They are frequently handled differently on different computers — but not always obviously so.

Algebraic Operators

Again, no problem. All are the same. The hyphen is used for negation as well as subtraction. Strings can be concatenated with the "+" sign.

Math Functions

All math functions are directly transferrable. The DEF FN, which allows you to create your own functions, is so weak on the Apple that it may be translated directly. About the only function that requires explanation is SGN. This returns the sign of the number (not its Sine): -1 if it's negative, 0 or 1 if it's positive. To simulate Y=SGN(X), use Y= (X<0) - (X>0).

The other area that needs to be explained is that of random numbers. The Apple's RND(1) is our RND(0); both produce random decimals. (And the Apple's RND(0) displays the most-recently chosen random decimal. It can be duplicated only by copying the most recent TRS-80 RND(0) into a variable and then using that.) The Apple has no RND(X). In order to display random integers between 1 and X inclusive, it uses Y= INT (RND(1)*X +1). You can replace the RND(1) with RND(0) or replace the entire expression with Y=RND(X).

System and Utility

FRE(0) on the Apple returns the amount of memory available to the user. Translate it into MEM. If you are specifically interested in the amount of string memory, use FRE(X$).

Now, the memory available to you on the Apple is set by HIMEM: and LOMEM:. They establish boundaries on the area that the program can use. You will frequently see HIMEM:8192. This sets the memory limit just below the high-resolution page one, so that a program does not overlap video memory. To translate it, ignore it but set the memory when you enter BASIC. Because Applesoft programs begin at 800H, add about 18000 (or 28000 for disk systems) to the number used in the program.

When you enter BASIC, you should also set the number of disk buffers that the Apple's MAXFILES requires. However, ignore any command with MON, which duplicates output to the disk and the screen.

Peeking and Poking

These commands are strange on the Apple. You may use either a positive number or its negative equivalent (difference from 65536); that means that POKE 49168,0 is equivalent to POKE -16368,0. This explains a lot; the high numbers are used because the Apple's ROM begins at D000H and continues up to FFFFH, and the pokes are to pressure points within it.

A second place pokes are used is to establish a machine language program — in *low* memory. The 256 bytes beginning at 768 are unused by the Apple, so most sound routines and other short subroutines are stuck there. A third place is on "page zero," the first 256 bytes; POKE 32,5 indents all text 5 spaces, for example.

Do not translate any pokes you do not understand or cannot find the equivalent of. They could do some pretty horrible things.

CALL and USR

The USR(X) command is just like our USR(X). It passes values back and forth from a machine language subroutine. Unless you know exactly how to translate the routine into Z80 machine language, see if there is a BASIC equivalent that will do the same thing a little slower or see if you can do without it entirely.

CALLs are used much more often because they can use ROM routines to do things quickly. In order to help you more easily translate common CALLS, a short table

is presented:

Apple CALL	Function	Translation
CALL -936	clears the screen	CLS
CALL -958	clears text to bottom	PRINT CHR$(31)
CALL -875	clears entire line	PRINT CHR$(29)
CALL -868	clears line to right	PRINT CHR$(30)
CALL -678	wait for ENTER	INPUT X$

Debugging Tools

STOP and CONT may be directly translated. The Apple's TRACE command can be substituted by TRON, and NOTRACE becomes TROFF.

Program Files

One Apple program can load another by using either a CHR$(4) or an embedded Control-D. In either case, the line may be translated. If you see something like:

10 PRINT CHR$(4);"RUN THE GAME" or
10 D$="" : REM CTRL-D
20 PRINT D$;"RUN THE GAME"
convert it into:
10 RUN "THEGAME/BAS"

The lengths of file names must be changed. In addition, drive specifications must be used. The Apple's boot drive is numbered 1. All Apple disks contain DOS, so a second drive is not always needed; however, it is numbered 2. The spec is shown by a ,Dx, where x is the drive number. For example:

10 PRINT CHR$(4);"LOAD THIS PROGRAM,D2"
becomes:
10 LOAD "THISPROG/BAS:1".

The Apple allows a directory of the disk to be shown from within a program: PRINT CHR$(4);"CATALOG" can be replaced with the equivalent of TRSDOS 1.3 DISK BASIC's CMD"D:0". It also allows machine language files to be easily saved, loaded, and run. ("B" stands for binary.) PRINT CHR$(4);"BRUN THIS PROGRAM" can be replaced by CMD"I","THISPROG/CMD", which returns to TRSDOS 1.3 and executes THISPROG. BSAVE can be done similarly. Its parameters include the starting address of the binary file and its length; an example is PRINT CHR$(4);"BSAVE PICTURE DATA, A$2000, L126", in which the $ indicates a hexadecimal value. It can be simulated by CMD"I",X$ with X$="DUMP PICTURE/DAT (START=....,END=....,TRA=....)" and with the adjusted parameters filled in. BLOAD can be simulated with the CMD"L" statement. It is frequently used to load sound routines or shape tables.

Text Files

Differences in the structure of text files are usually among the hardest to overcome.

Sequential Files

The Apple opens a sequential file when it encounters a PRINT CHR$(4);"OPEN FILENAME". You may also see a drive specification with the name. Once a file is opened, it can either be written to or read from. The Apple is told to do this by a 20 PRINT CHR$(4);"WRITE FILENAME" or 20 PRINT CHR$(4);"READ FILENAME". Thereafter, any INPUT X$ reads from the disk file and a PRINT X$ writes to it. One advantage of this arrangement is that the file does not need to be closed and reopened when switching from WRITE to READ. On the other hand, the easiest way to accomplish the conversion to your TRS-80 is to reopen the file. To open a file for output (a "WRITE"), use:

10 OPEN "O", 1, "FILENAME/TXT"

The "1" indicates the buffer number. Then change all PRINTs within the block to PRINT#1s in order to accomplish the writing. To convert a read, use 10 OPEN "I", 1, "FILENAME/TXT", and change all INPUTs to INPUT#1s. Translate the PRINT CHR$(4);"CLOSE FILENAME" as CLOSE.

The Apple, like the TRS-80, also handles appending by reopening the sequential file. A 10 PRINT CHR$(4);"APPEND FILENAME" becomes 10 OPEN "E", 1, "FILENAME/TXT".

Unfortunately, there is one command that is not directly translatable. POSITION, Rx moves the file pointer forward X fields. A clumsy but workable solution is to simply read the next X records.

The command EXEC executes the commands in the text file. If the first record is PRINT "THE TEXT FILE IS BEING EXECUTED", that message will be printed. It, too, cannot be translated.

Random-Access Files

The only difference in opening random-access files on the Apple is that the length of the records must be specified. 10 PRINT CHR$(4);"OPEN RANDOM,L40" sets the fields to have a length of 40 bytes. Unfortunately, the length parameter can be any positive integer. Chances are, however, that you will be able to use the standard 10 OPEN "R", 1, "RANDOM/TXT",L40 to open a file with the same-sized records. You must also be sure to include a FIELD statement to divide the records into fields. Something like FIELD 1,40 AS NAME$ serves admirably to set those 40 bytes to be NAME$.

READs and WRITEs are also handled differently. They are succeeded by the Rx we saw earlier. This tells the Apple to READ from or WRITE to random-access record number X. Typically, the number of records is stored in R0. Thereafter, PRINT CHR$(4);"WRITE RANDOM,R"X will write to record #X. Note that X is outside of the quotes.

Both READs and WRITEs on the TRS-80 are accomplished with the use of the file buffers and a record pointer. If X% is the number of the record to be retrieved, then GET 1,X% reads it. It then becomes NAME$ (because it was fielded that way) and can be manipulated. PUT 1, X% writes to that record. The basic idea of the two sets of commands is the same, but the syntax of the commands are different.

Where to Find Apple Software

Two good Apple-oriented magazines are *Softalk* and *Nibble*; another is *Incider*. They concentrate on BASIC.

Well, in a nutshell, there you have it. The best aid to translating a program from one dialect to another is understanding how it works. I hope I have given you the tools that will allow you to use a whole new world of software.

LOGICAL SYSTEMS AND YOUR MODEL 4
CREATE A FUNCTIONAL COMBINATION

LS-FED II 6.x (FILE EDITOR) — The ultimate "Zapping" utility with Visual Disassembler built in. File and/or track/sector oriented. Catalog. #L-30-012.
only $49.00 plus $3.00 Shipping & Handling

LS-FM 6.x (FILE MANAGER) — Conditional Parameterized, Move, Kill, Cross Examine, Create Secondary Action Files and Much, Much More. Catalog #L-30-051.
only $49.00 plus $3.00 Shipping and Handling.

LS-TBA 6.x (The BASIC Answer) — TBA'S 5 Pass Text Processor allows structured programming for TRSDOS 6.x Basic. 14 character variables and Local variables (PSUEDO). No Line Numbers, use Labels instead, with Cross Reference System.
Catalog #L-21-011 only $79.00 plus $4.00 Shipping & Handling

LS-HELP 6.x (BUILT IN 6.x TUTOR) — The Handiest Utility a New Model 4 owner could possibly have. Provides almost all the information needed to use the powerful features of TRSDOS 6.x.
Catalog #L-30-061
only $29.00 plus $3.00 Shipping & Handling.

All of the above products are also available for the TRS-80™ Models I and III running under the LDOS 5.1 Operating System.

Contact Logical Systems, Inc. for a free detailed catalog containing these and many other products.

* Prices and Specifications subject to change without notice.
* TRSDOS™ and TRS-80™ are Trademarks of Tandy Corp. LDOS, LS-FED II, LS-FM, LS-TBA, LS-HELP are all products and trademarks of LSI. (The authors of TRSDOS™ 6.x)

LOGICAL SYSTEMS INC.

LOGICAL SYSTEMS, INC. 8970 N. 55TH ST. MILWAUKEE, WI 53223 (414) 355-5454

CSAVE 100

Saving Model III/4III programs to be read by a Model 100. No hardware or terminal program needed!

Model III/4III and Model 100

John and Aileen Cornman

When a new machine like the Model 100 comes along, how do you get your Model III BASIC programs transferred over? The Model III can CSAVE at 1500 baud, which is compatible with the Model 100, but the format is different.

If you happen to have RS-232C on your Model III, with the right cables, a terminal program, and a little work, you can set up a communication link that allows you to transfer ASCII files of BASIC program text to the Model 100. (The November, 1983 issue of *TRS-80 Microcomputer News* has an article on doing just that. —Ed.) But, rather than buying RS-232C, you can use the accompanying programs to make your Model III CSAVE programs in a format that is CLOADable on the Model 100.

SAV100 is an assembly language program for the Model III that writes BASIC programs on tape using the Model 100 format. After installing SAV100 in the highest 791 available memory locations of your Model III, you can CLOAD Model III BASIC tapes, then use SAV100 to create the Model 100 tapes.

Installing the Program

"Model 100 CSAVE Loader" (Listing 1) is the BASIC program you must run to install SAV100 in your Model III. The Loader automatically adjusts the machine language instructions of SAV100 in accordance with the amount of memory you have. This feature makes it usable on 16K, 32K and 48K machines without modification. The Loader helps you find mistyped hex codes in the DATA statements and computes a checksum to insure their accuracy. Follow these steps to use the Loader:

1. When you turn on your Model III, you must answer the "Memory Size?" question according to the amount of memory you have:

Machine	Memory Size?
16K	31976
32K	48360
48K	64744

2. Run the Loader program. If you made an error in your answer to the memory size question, the program reminds you of the correct responses and asks you to try again.
3. The Loader asks you to wait while it reads the hexadecimal codes in the DATA statements and loads them into memory.
4. If the Loader finds an unacceptable hexadecimal character in a DATA statement, it shows you the bad character pair and tells you the linenumber and pair number that need to be corrected before you rerun the Loader. There are 18 pairs of characters in every DATA line except the last, which has 17.
5. While all of the hex codes may be acceptable, they may still be incorrect, so the Loader informs you if the hex codes in the DATA statements do not add up to the predetermined checksum total. With this kind of error, there is no further information the Loader can give you to help find the incorrect code. You must recheck your DATA statements against Listing 1 to find the error before you rerun.
6. If no errors are detected, the Loader tells you the entry address to use with the SYSTEM command when you run SAV100.
7. You are now ready to CLOAD a BASIC program that you want to transfer to the Model 100.

Using SAV100

1. CLOAD a Model III BASIC program. If you have tapes recorded at both 500 and 1500 baud, set the right baud rate before you CLOAD by using these POKE commands:

POKE 16913,72 to set the high rate of 1500 baud
POKE 16913,0 to set the low rate of 500 baud

SAV100 always writes at 1500 baud and will not affect the baud rate you have set.

2. Enter the command SYSTEM.
3. At the asterisk prompt, enter:

/31977 for a 16K machine

/48361 for a 32K machine
/64745 for a 48K machine
Remember to type the slash in front of the number and press enter at the end.

4. The SAV100 title screen appears and asks you to specify a one- to six-character name for the program. This is the name you will use when you CLOAD the tape on your Model 100, so be sure to follow Model 100 file naming conventions by making the first character a letter. (SAV100 does not check this.)

5. SAV100 prompts you to press enter when the recorder is ready.

6. After the tape is written, SAV100 asks you if you would like to make another copy. If you answer with a "Y," you return to step five. When you answer with an "N," you go on to step seven.

7. The BASIC READY prompt reappears. You may return to step one, ready to CLOAD another BASIC program tape. For some reason we cannot explain, BASIC is likely to respond with an OM (out of memory) error message to whatever command you enter next. We found that no harm is caused by this condition, but to clear the error message, enter a command such as ?MEM.

Saving SAV100

You may load a working copy of SAV100 at any time by simply running the BASIC Loader program, but if you use it often, it would save time to have a machine language SYSTEM tape that could be loaded directly into your Model III. If you have a Debug or Monitor program that allows you to write SYSTEM tapes, you will need the following information:

	START ADDRESS		END ADDRESS		ENTRY POINT	
Mem.	Hex	Decimal	Hex	Decimal	Hex	Decimal
16K	7CE9	31977	7FFF	32767	7CE9	31977
32K	BCE9	48361	BFFF	49151	BCE9	48361
48K	FCE9	64745	FFFF	65535	FCE9	64745

If you ever add more memory to your computer, you will need to use the Loader program again to tailor SAV100 to the new high memory addresses.

Model III to Model 100 Conversion Guidelines

Some Model III programs may run on your Model 100 with no modification, but most will need tailoring. SAV100 does some of the work for you by making these changes:

1. SET and RESET become PSET and PRESET, so the statements will still work, but they will only use the upper left 48 by 128 dots of the Model III screen.

2. MEM becomes FRE(1) to perform the same function.

3. USR becomes CALL, but Model 100 machine language calls have a different format. Z-80 machine language subroutines that do not rely on the Model III memory map or port-oriented I/O can be used on the

Model 100 only if they are restricted to the 8080-compatible subset of instructions.

4. TIME$ becomes TIME$+DATE$ and will return the same result.

5. The following Model III BASIC words are preceeded by a REM statement because they have no Model 100 counterparts: CMD, RANDOM, TRON, TROFF, FIELD, GET, PUT, LSET, RSET, MKD$, SYSTEM, DELETE, AUTO, FN, POINT, CVI, CVS, CVD, MKI$, MKS$. Most of these are rarely used as program statements and should cause few problems. The Model 100 manual suggests a routine to take the place of the RANDOM function. There is no substitute for the POINT statement.

The following statements translate to identical BASIC keywords on the Model 100, unchanged by SAV100. They may function differently, so check them carefully.

IF sometimes requires a THEN in Model 100.

LINE draws a line on Model 100 screen.

OUT and INP will need new port numbers.

OPEN, CLOSE and EOF need tailoring for Model 100 files.

MERGE must have an ASCII file in the Model 100.

NAME must use the Model 100 naming conventions.

KILL can only refer to Model 100 RAM files.

DEF is used only in DEFINT, DEFSTR, DEFSNG and DEFDBL statements.

POKE and PEEK need special attention to the addresses used.

PRINT@ and POS need tailoring to the Model 100 screen size.

VARPTR may return different values on the Model 100.

ERR returns an error code directly. There is no conversion formula.

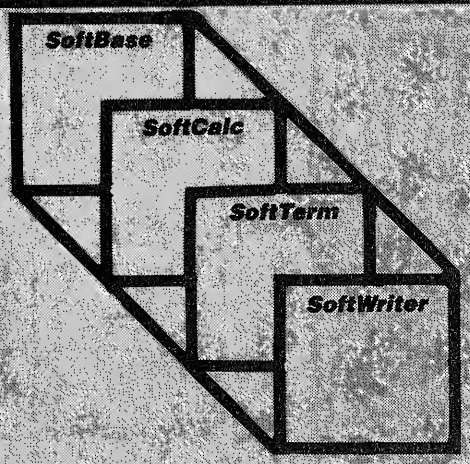
RND only returns values between 0 and 1.

Listing 1 — Model 100 CSAVE Loader

```
10 CLS:DEFINTA-Z
20 PRINTTAB(21)"Model 100 CSAVE Loader"
30 PRINTTAB(15)"by Structured Software S
ervices":PRINT
40 PRINT:PRINT"Loading SAV100 into memor
y, please wait..."
50 GOSUB 550   'CHECK MEMORY SIZE AND SE
T UP ADDRESSES
60 FORI=1TO791
70      READ HX$                    'GET A P
AIR OF HEX DIGITS
80      D$ = LEFT$(HX$,1)           'LOOK AT
 THE LEFT DIGIT
90      IF D$ = "*" THEN 210        '* ME
ANS ADDRESS ADJUSTMENT
100     GOSUB 500                   'CONVERT
 HEX DIGIT TO DECIMAL
110     IF D = -1 THEN 400          '-1 IS F
LAG FOR BAD HEX CHAR.
120     DC = D * 16                 'DC GETS
 VALUE OF LEFT DIGIT
130     D$ = RIGHT$(HX$,1)          'LOOK AT
 THE RIGHT DIGIT
140     GOSUB 500                   'CONVERT
 RIGHT DIGIT
150     IF D = -1 THEN 400
160     DC = DC + D                 'DC = TH
E DECIMAL CONVERSION
170     CS! = CS! + DC              'ADD IT
TO THE CHECKSUM
180     POKE PA,DC                  'LOAD IT
 INTO MEMORY
190     IF PA < 32767 THEN PA = PA + 1 '
ADVANCE POKE ADDRESS
200     GOTO 300                    'READY F
OR NEXT HEX PAIR
210     D$ = RIGHT$(HX$,1)          'D$ = RI
GHT DIGIT OF *-PAIR
220     GOSUB 500
230     IF D = -1 THEN 400
240     CS! = CS! + D               'ADD TO
CHECKSUM
250     DC = BA + D                 'BASE AD
DRESS + DISPLACEMENT
260     GOTO 180                    'GO POKE
 ADJUSTED ADDRESS
300 NEXT
310 IF CS! <> 76724 THEN 350        'VERIFY
```

```
CHECKSUM TOTAL
320 PRINT:PRINT"SAV100 has been successf
ully loaded."
330 PRINT:PRINT"Entry address for SYSTEM
 command is:";PA!
340 PRINT:PRINT"You may now load BASIC p
rograms and CSAVE in Model 100 format."
: PRINT : END
350 PRINT:PRINT"Checksum error."
360 PRINT:PRINT"Please correct contents
of DATA lines 800 - 886 and rerun." : PR
INT : END
400 LN = INT((I-1) / 18)          'RELATIV
E LINE # OF ERROR
410 CH = I - (LN * 18)            'BAD CHA
RACTER PAIR #
420 LN = 800 + LN * 2             'ACTUAL
DATA LINE #
430 PRINT:PRINT"Invalid hex character: "
;HX$
440 PRINT:PRINT"Please correct pair #";C
H;"in line";LN;"and rerun."
450 PRINT : END
500 D = ASC(D$)
510 IF D > 47 AND D < 58 THEN D = D - 48
 : RETURN '0-9 OK
520 IF D > 64 AND D < 71 THEN D = D - 55
 : RETURN 'A-F OK
530 D = -1 : RETURN      'BAD HEX CHARACT
ER
550 BA = PEEK(16562)    'GET BASE ADDRESS
560 IF BA = 252 OR BA = 188 OR BA = 124
THEN 570 ELSE 620
570 IF PEEK(16561) <> 230 THEN 620
580 PA! = BA * 256 + 233          'PA! = ST
ART ADRESS FOR POKING
590 IF PA! > 32767 THEN PA = -1 * (65536
 - PA!) ELSE PA = PA!
600 RETURN  'PA ADJUSTED IF PA! > 32767
620 PRINT:PRINT"Before running SAV100 Lo
ader, you must answer the Memory Size?"
630 PRINT"question according the the fol
lowing table:" : PRINT
640 PRINT"Machine     Memory Size?"
650 PRINT"  16K          31976"
660 PRINT"  32K          48360"
670 PRINT"  48K          64744"
680 PRINT:PRINT"Please press the orange
reset button and answer the"
690 PRINT"Memory Size? question again be
fore rerunning this program."
700 GOTO700
800 DATA CD,C9,01,DD,21,00,*3,21,18,3C,2
2,20,40,21,5C,*2,CD,1B
802 DATA 02,21,50,3C,22,20,40,21,6C,*2,0
6,03,CD,1B,02,10,FB,CD
804 DATA 58,*1,CD,A4,*1,DA,19,1A,3A,11,4
```

```
2,32,5B,*2,3E,48,32,11
806 DATA 42,21,B4,*2,CD,1B,Ø2,CD,1B,Ø2,C
D,49,ØØ,CD,C4,*1,Ø1,ØØ
808 DATA Ø8,CD,6Ø,ØØ,CD,F5,*1,CD,F8,Ø1,2
1,D8,*2,CD,1B,Ø2,CD,1B
81Ø DATA Ø2,CD,49,ØØ,FE,59,28,D5,FE,79,2
8,D1,3A,5B,*2,32,11,42
812 DATA C3,19,1A,E5,16,ØØ,Ø1,ØØ,ØØ,21,E
9,43,7E,23,B6,28,38,Ø3
814 DATA 23,Ø3,23,Ø3,23,Ø3,7E,B7,2Ø,Ø4,Ø
3,23,18,EC,D6,8Ø,3Ø,Ø4
816 DATA Ø3,23,18,FØ,32,82,*1,DD,7E,ØØ,F
E,8Ø,3Ø,F2,5F,E5,21,8Ø
818 DATA *3,19,7E,FE,8Ø,38,Ø1,23,7E,B7,2
8,Ø4,Ø3,23,18,F8,E1,18
82Ø DATA DB,ED,43,FE,*2,E1,C9,CD,1B,Ø2,2
1,F6,*2,Ø6,Ø6,CD,4Ø,ØØ
822 DATA D8,48,Ø6,ØØ,Ø9,3E,Ø6,91,28,Ø7,4
7,3E,2Ø,77,23,1Ø,FC,3E
824 DATA EØ,77,C9,21,ØØ,ØØ,Ø6,ØA,ØE,ØØ,C
D,87,Ø2,21,F6,*2,3E,D3
826 DATA CD,64,Ø2,7E,CD,ED,*1,23,1Ø,F9,3
E,AA,Ø6,Ø6,CD,ED,*1,1Ø
828 DATA FB,79,ED,44,CD,64,Ø2,C9,57,81,4
F,7A,CD,64,Ø2,C9,21,ØØ
83Ø DATA ØØ,ØE,ØØ,CD,87,Ø2,3E,8D,CD,64,Ø
2,21,E9,43,7E,23,B6,2B
832 DATA 2Ø,Ø7,79,ED,44,CD,64,Ø2,C9,Ø6,Ø
4,7E,CD,ED,*1,23,1Ø,F9
834 DATA 7E,B7,2Ø,Ø6,CD,64,Ø2,23,18,EØ,F
E,8Ø,3Ø,Ø6,CD,ED,*1,23
836 DATA 18,EC,D6,8Ø,32,36,*2,DD,7E,ØØ,F
E,8Ø,3Ø,EE,5F,16,ØØ,E5
838 DATA 21,8Ø,*3,19,7E,FE,8Ø,3Ø,Ø5,3E,8
E,CD,ED,*1,7E,B7,28,Ø6
84Ø DATA CD,ED,*1,23,18,F6,E1,23,18,CØ,Ø
Ø,4D,6F,64,65,6C,2Ø,31
842 DATA 3Ø,3Ø,2Ø,43,53,41,56,45,ØD,62,7
9,2Ø,53,74,72,75,63,74
844 DATA 75,72,65,64,2Ø,53,6F,66,74,77,6
1,72,65,2Ø,53,65,72,76
846 DATA 69,63,65,73,ØD,ØD,ØD,45,6E,74,6
5,72,2Ø,31,2Ø,74,6F,2Ø
848 DATA 36,2Ø,63,68,61,72,61,63,74,65,7
2,2Ø,7Ø,72,6F,67,72,61
85Ø DATA 6D,2Ø,6E,61,6D,65,3A,2Ø,Ø3,ØD,5
Ø,72,65,73,73,2Ø,45,4E
852 DATA 54,45,52,2Ø,77,68,65,6E,2Ø,72,6
5,63,6F,72,64,65,72,2Ø
854 DATA 69,73,2Ø,72,65,61,64,79,2E,ØD,5
7,72,69,74,65,2Ø,61,6E
856 DATA 6F,74,68,65,72,2Ø,63,6F,7Ø,79,3
F,2Ø,28,59,2Ø,6F,72,2Ø
858 DATA 4E,29,ØD,ØØ,ØØ,ØØ,ØØ,ØØ,ØØ,EØ,D
5,ØØ,ØØ,8Ø,81,B5,B4,BØ
86Ø DATA ØØ,Ø4,82,83,84,85,86,87,88,89,8
A,8B,8C,8D,8E,8F,91,ØB
862 DATA 1Ø,16,1B,1E,23,92,93,94,95,96,9
7,99,28,2E,32,9A,9B,9C
864 DATA BC,BD,36,3B,9E,4Ø,AØ,A1,A2,A3,A
4,A5,A6,47,4E,A7,A8,A9
866 DATA BF,CØ,C1,53,C2,C3,B9,C4,C5,C6,C
7,56,7C,5C,C9,CD,CE,CF
868 DATA DØ,D1,D2,D3,D4,D5,D6,DC,DD,DE,D
F,EØ,E1,E2,E3,E5,E6,E7
87Ø DATA E8,E9,EA,EB,EC,ED,EE,61,65,69,E
F,FØ,F1,6D,72,77,F2,F3
872 DATA F4,F5,F6,F7,F8,F9,FA,FC,FD,FE,F
F,8E,8E,8E,8E,43,4D,44
874 DATA ØØ,52,41,4E,44,4F,4D,ØØ,54,52,4
F,4E,ØØ,54,52,4F,46,46
876 DATA ØØ,A1,53,54,52,ØØ,A1,EØ,ØØ,A1,5
3,4E,47,ØØ,A1,44,42,4C
878 DATA ØØ,46,49,45,4C,44,ØØ,47,45,54,Ø
Ø,5Ø,55,54,ØØ,4C,53,45
88Ø DATA 54,ØØ,52,53,45,54,ØØ,53,59,53,5
4,45,4D,ØØ,44,45,4C,45
882 DATA 54,45,ØØ,41,55,54,4F,ØØ,46,4E,Ø
Ø,5Ø,4F,49,4E,54,ØØ,E2
884 DATA 28,31,29,ØØ,43,56,49,ØØ,43,56,5
3,ØØ,43,56,44,ØØ,4D,4B
886 DATA 49,24,ØØ,4D,4B,53,24,ØØ,4D,4B,4
4,24,ØØ,AA,2B,AB,ØØ
```



SoftPac

Integrated Software for your TRS-80

Meet SoftPac — the remarkable new software package that puts more raw power at your fingertips than anything yet created for the Radio Shack computer. Integration is the key. SoftPac actually combines word processing, information management, spread sheet calculations, and communications into one fully integrated system. Data is transmitted from one program to another at the touch of a key. Don't worry, SoftPac doesn't cheat by using disk files, it transfers data through memory. This means that SoftPac is fast and easy to use! Perhaps the single reason why you should own SoftPac is that all updates to the package are free! The SoftPac program package includes the Master and Backup diskettes, and over 200 pages of documentation. This includes hundreds of models and examples! Just look at all those features in each SoftPac program!

SoftWriter

Insert characters, words, lines, paragraphs, screens & more. Delete characters, words, lines, paragraphs, screens & more. Variable Margin widths from 1 to 255 and Right Justification. Simple Arrow key movements with automatic text scrolling. Automatic Line Centering and Word Wraparound supported. Block Copy of text from 1 to 1000 characters instantly. Block Copy and Deletion of text from 1 to 1000 characters. Global Find and Replace commands with Wild Cards supported. Automatic Key Repeat and two control keys for command access. Whoops feature to prevent deletion of valuable text. Transmit and Receive text from any other SoftPac program. Save part or all of a document in the SoftWriter memory. Merge two or more files together to form one document. Complete spelling checker to correct any misspelled words. Underline, Emphasize, Subscripts, Superscripts, Correspondence, variable character sizes, form feed, page numbers, left margin, right margin, page length, line length, all printers supported. Complete printer control codes allow you to cut & paste text. Automatically counts the number of letters, words, & sentences. Split screen viewing of any file on your data diskette. Type ahead of any keyboard character or command supported.

SoftTerm

Variable Screen Width from 1 to 255 with Word Wraparound. Printer Output supported with High Speed Printer Spooler. Carriage Return and Line Feed Suppression command supported. Upper/Lower command converts all uppercase text to lowercase. Echo Keyboard command for incoming and outgoing characters. Input Buffer and Output Buffer command to a 10,000 byte buffer. Place text from other SoftPac programs in the SoftTerm buffer. Transmit text to other SoftPac programs from SoftTerm. Send all ASCII characters and re-program the keyboard. Complete translation tables for incoming, outgoing, & printer. Review data that has scrolled off the top of the screen. Communicate with any system including a EBCDIC-coded system. Change the Baud Rate, Parity, Stop Bits, and Word Length. Automatic count and display of all transmit and receive errors. Automatic dialing, Hang Up, Directory, and Pause features. Complete Modem & RS-232 test command for hardware analysis. Transmit and Receive Basic and Binary files from any system. Save all current SoftTerm settings to your data diskette. Nulls supported for incoming, outgoing, and printer characters. Several supplied setting files for your printer and keyboard. Special commands allow buffer transfer and Cold Startups.

SoftCalc

Text and Character entry into any position for labeling. Easy cursor movement using the Arrow keys and Control keys. Block cursor movement to each corner of the spread sheet. Automatic key repeat and type ahead of any character typed. Replication and Forecasting command for value prediction. Deletion of Labels or Numbers in any Column, Row, or Position. Sorts any Column or Row in Ascending or Descending order. Print any portion of the spread sheet to any standard printer. Move a Label, Number, Position, Column, or Row instantly. Calculate the Total of any Column or Row on your spread sheet. Set up financial models with cross referencing of values. All standard math functions supported including Trig functions. Many Engineering and Scientific functions fully supported. Integrate & merge SoftCalc data into any other SoftPac program. Instant scrolling of the screen during Row or Column movement. High resolution screen supports 8 column width & 14 Row depth. Instant on-screen graphs of your data in Bar or Pie plots. Complete error trapping routines prevent loss of any data. Simple operation with complete prompting on the command line.

SoftBase

Set up files using customized forms generated by the data base. Use Multiple forms in any portion of the SoftBase program. Use all or part of a form in a file with control commands. Select the Full Screen Text Editor mode for data entry. SoftBase does not limit you to a specific field or entry size. Read any file using the file name or the Wild Card feature. Correct any file instantly using the Full Screen Editor mode. Delete any file from the data base with the touch of a key. Read Multiple files on screen and print each file to a printer. Search for a file by the letter, word, sentence, and more. Locate any file in the data base in under 1 second at any time. Review selected portions of each group of 10 files on screen. Print one, all, or selected files to any standard printer. Insert text from any SoftPac program into a SoftBase file. Transfer text from any SoftBase file to other SoftPac programs. Print mailing labels and merge SoftBase files with SoftWriter text to generate form letters using any portion of the file. Memory command displays available memory and unused files. Complete Error Trapping routines and High Speed operation.

SoftPac V3.0 Disk & Manual $299.00
SoftPac V3.0 Manual & Demo 25.00
Available for any Model I, III, & 4.
Dealer Inquiries are Welcome.

Visa and C.O.D. orders welcome.

Prices & Availability subject to change

#17

TRS-80 is tm of Radio Shack, a Tandy Corp

918-749-6211

Softronics Computer Systems
2300 E. 14th Suite 201
Tulsa, Oklahoma 74104

The future of software design

Key issues facing today's software developers

For all readers

William Gates, Microsoft Corp.

Software, after years of taking a backseat to hardware, has finally come into its own. Today we see general acknowledgement of the importance of the role that software plays. Software is the bridge between the machine and the user, the tool that brings the power of the computer to the user. And it is software that is defining today's key information issues.

Instead of the emphasis of past years on building better and more powerful machines, the emphasis now is on how to harness the full power of the existing hardware through improved software design. The promise is that the existing machines could do the job much better (more easily, more efficiently) if software was better designed.

This promise, in turn, leads straight into several key issues that are facing software developers today. Because what, exactly, constitutes a better design? Of the various approaches that software design can take, which will be most effective in helping users access the full potential of their machines?

There are currently five major issues facing software developers. None has easy answers. The stand that each of the major players in the field chooses to take on these issues (and the degree to which the ultimate judge, the user marketplace, accepts the stand taken) will determine the future direction of software design.

A great deal of money is going to be invested in these choices. The cost of developing a fully-integrated family of applications is enormous. Apple talks of investing $50 million to develop a complete applications family; Xerox views the job in terms of hundreds of man-years. Therefore, each software developer is going to have to take a good, hard look at each of these issues, and make its choice with great care. A wrong choice will be costly at best; at worst, it could spell financial disaster.

In this article, we'll examine each of today's key software issues, analyze the pros and cons of the possible choices within each issue, and hazard some guesses as to which directions will turn out to hold the key to the software packages of the future.

Each software developer is going to have to take a good, hard look at each of these issues . . . a wrong choice will be costly at best; at worst, it could spell financial disaster.

Issue #1: Integration

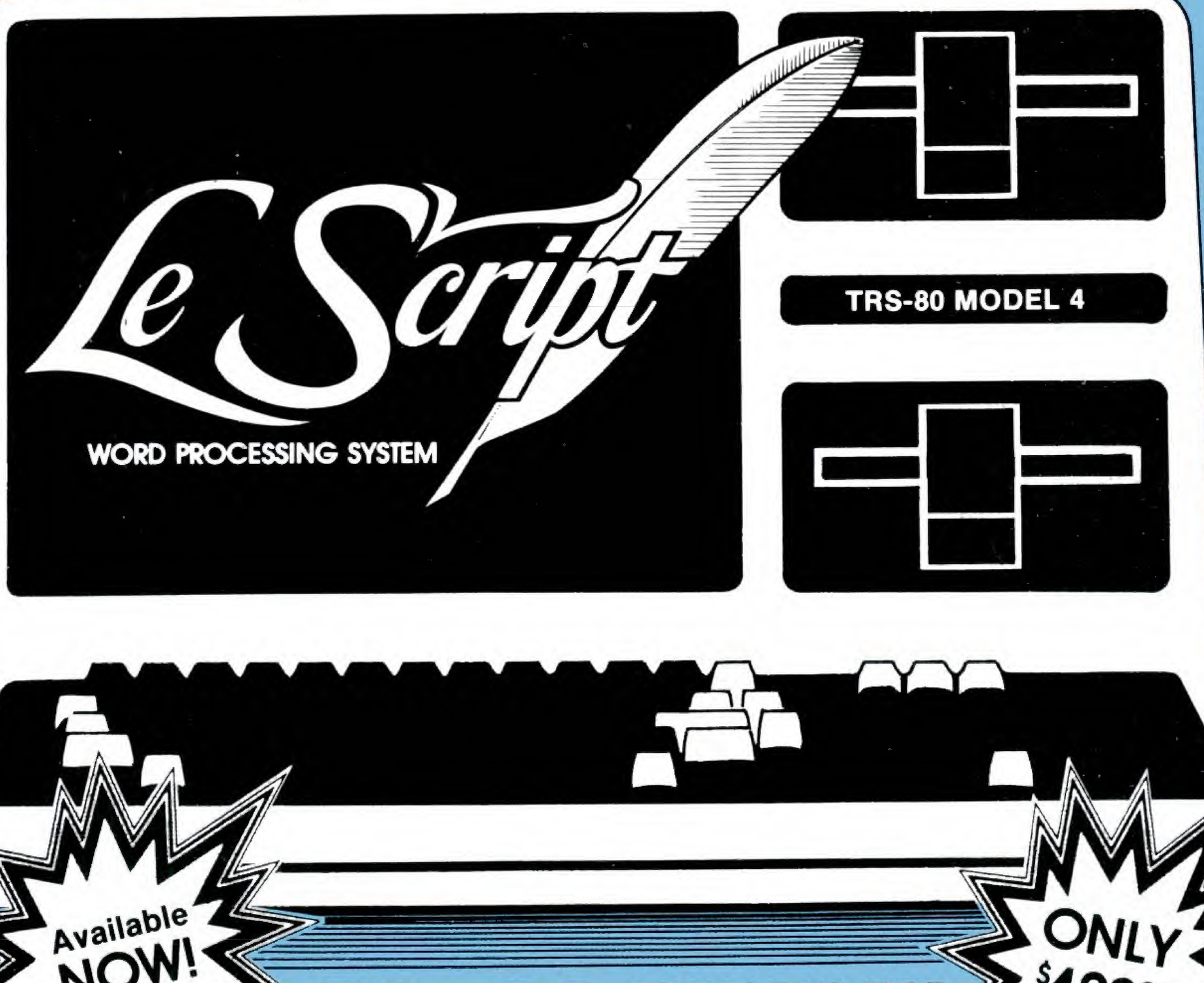
Integration has been a byword in the software industry for some time. But the issue here is not superficial integration. We are not talking about taking our products and calling them by similar names. We are not even talking about moving the data back and forth between the products through some sort of low-level numeric description, where special commands must be given each time the user wants to move data from one application to another.

Such an approach, although better than no integration at all, presents the user with two major problems. First, special commands take considerable time and effort, both in the initial learning and in their application each time the data is to be moved. Worse yet, with this type of integration, key information about the data is lost. Take sales data, for example. In a particular application, the user may have described sales by time period (daily, weekly, or monthly), by sales unit (sales rep, product line, or division), and by the form in which he wants to print it. With today's level of integration, if he tries to move this data from one application to another, he generally will lose some of these important descriptors. The data will be devoid of its full structure.

The two key features of real integration are that it must capture all data descriptors and that it must be automatic. That is, to get two applications to work together, there should be no need to continually move the data back and forth manually. If, for example, a user needs to combine data from his Balance Sheet and his Income Statement to do his monthly reports, he should be able to specify what data he wants the reports to include

See us at
SOFTCON
La Superdome
New Orleans
Tuesday-Thursday
February 21-23, 1984
BOOTH P-2063

TRS-80 MODEL 4 Word Processing Program



Available NOW!

ONLY $129.95

80x24 DISPLAY

- Works with **TRSDOS 6** or any **Model III DOS.**
- **90K TEXT BUFFER** when expanded memory is installed.
- Built-in **FORM LETTER** and **MAIL-MERGE** capabilities.
- Performs **PROPORTIONAL-SPACE** right-margin justification on over 60 different printers (all drivers included **FREE**).
- Same powerful editing features as **ZORLOF II,** with dozens more features added.
- LeScript also available to run on **MODEL I, MODEL III, LNW-80, PMC-80,** and **MAX-80** (80x24 display).

FREE

- 30-Day Support to Registered Owners
- Subscription to LeScript Newsletter

LeScript costs $200 less than competing products matching its capabilities.

VISA MasterCard

CALL 1-305-259-9397

ANITEK

ANITEK SOFTWARE PRODUCTS
P.O. BOX 361136
MELBOURNE, FL 32936 #18

FREE SHIPPING WITHIN THE U.S.; OUTSIDE THE U.S. ADD $10.00 FOR SHIPPING; FLORIDA RESIDENTS ADD 5% SALES TAX. ALL ORDERS PREPAID BY CHECK, MONEY ORDER, CREDIT CARD OR C.O.D.

and what format he wants it printed in. The rest should be automatic (graphs, charts and all) without any need to go back in and reinput or redescribe the data.

This, then, is how fully integrated software will work. But the big question is, how do we get there? Basically, there are two possible approaches: either we build one single application that does everything, or else we find better ways of moving data between separate applications.

The first approach has a definite appeal, in view of the fact that no one has yet developed a way of moving data between applications in a high-level form. But there are three significant drawbacks to the idea of building a single applications package that does it all.

First, there is the problem of specialized expertise. Even if one software developer had the expertise to build a complete set of generic applications (time scheduling, project scheduling, data base development, electronic spreadsheets, and the like), it would be impossible to find a single vendor who had the expertise to build all the necessary vertical applications. And vertical packages specific to different professions or companies are going to be a major segment of the software market. This need, then, points to the importance of developing an approach to integration that lets different parties with specialized types of expertise come in and provide specific vertical applications of the various packages.

A second problem with the approach of developing a single application that does it all is that it requires the selection of a single data structure. Since a data structure that is ideal for one application may be clumsy and inefficient for another, the net effect of this approach is that it compromises individual applications. For example, an in-memory data structure that is well-suited to a spreadsheet application may be poorly suited for a data base package. In fact, it may be totally unusable. If a user wants to develop graphs from the data stored in all the separate cells of a spreadsheet, for example, and he has to move the cells around and give a special set of commands each time he needs a graph drawn (or, alternatively, find a macro string that will accomplish the same end), he's not going to be likely to use the application very frequently. Clearly, different applications require different data structures to make them easy to use.

The third difficulty with the single-application approach is that the command structure can easily become overstrained. The number of different commands and decision trees can become a significant problem.

For all of the above reasons, Apple and Microsoft are in agreement that the best solution is to have multiple products that can easily pass data back and forth. This doesn't mean that the products cannot be priced as a single package, or that they can't all be on the screen at one time. It does mean that they will be based on different data structures and will use different command structures.

Issue #2: User Interface

A second key decision area facing software developers today involves the development of standards for user interface issues. We do, today, have general agreement on some of these issues. For example, it is generally accepted that packages should include on-line "help" files, so users can immediately call up a piece of help text that is designed for the specific context in which they find themselves. Similarly, menus written in standard English and the use of full-sentence prompts are generally accepted. VisiCorp, for example, is moving away from the use of coded commands (/) and toward the use of English words.

The big issue today in the area of user interface is the introduction of a new element into the picture: graphics. Now graphics, to many people, implies the drawing of bar charts, isometric charts, and the like. But the graphics issue is, in reality, far broader than that.

The issue is really one of how we present data on the screen. So far, we

have been fairly confined in how we use the screen to present data. For a long time, we could only put characters (and monospaced ones, at that) in specific positions on the screen. This may not seem like a problem at first glance, but stop and think for a minute. If, every time you went to use a piece of paper, or a chalkboard, you had to take little letters and place them where you wanted them, wouldn't you find this approach to be restricting? You might find yourself using the paper or chalkboard a great deal less than you now do, when you have the freedom to put arbitrary images there in any form you want them.

The new graphic technology, with its use of pixels, bit-mapping and the like, is bringing this same richness to the computer screen. This ability to view the screen as a piece of paper, and to put arbitrary images on it, means that graphics are going to be used for a great deal more than just drawing graphs. The use of icons, for example, to tell the user what is happening, is a much more compact and compelling approach than using words to say the same thing. Cursor displays to show the user his position are another form of visual feedback. For example, when the user is deleting something, the screen could show scissors moving around the material being deleted. Even graphs and diagrams will be revolutionized by the new graphics technology, because the time and effort required to produce them will be significantly reduced. In fact, what the new graphics technology represents is a revolution in user interface.

The bottom line is that graphics are going to be a standard part of all computers. No machine that costs more than $1,000 will be without a built-in bit-map graphics screen. And the software analog of that hardware statement is that, one year from today, no decent application software family, no decent language family, and no decent operating system will be without extremely high-level support for this type of graphics capability. It will be no small task for the software developers to achieve this graphics integration, but it is a necessary task. Furthermore, the graphics

capability is not going to be in the form of an add-on package that the user goes out and buys after he has bought his computer. It will be part of the definition of the machine itself. As such, it will require very high-level primitives to allow the user to easily access the graphics capabilities.

As the above observations indicate, software developers are going to have to agree on some user-interface standards to allow the full power of this graphics revolution to be felt. First, they will need to develop some standards for incorporating the graphics capability into the machine. Apple is already moving in this direction with its development of a strong operating system as a foundation for such built-in features. Second, they will need to agree on some high-level operating system commands to make the graphics capabilities readily accessible to the user.

Issue #3: Data Storage Metaphors

Selection of the most appropriate data storage metaphor is one of the toughest issues facing the software industry today. Basically, this term refers to the way the user perceives the storage of data within the system. Take Apple's Lisa system, which is supposed to be capable of being learned in 20 minutes. Learning the spreadsheet application is going to be easy only for people who are used to working with formulas — people who like formulas, who understand them, and who understand how they can work together in an interdependent fashion. A data storage metaphor that is based on placing formulas in cells of a spreadsheet is never going to be easy for most people to learn regardless of how the system is dressed up with easy-to-remember icons, simple English commands, and so forth.

Xerox, on the other hand, uses a linear, document-oriented metaphor. It includes different types of frames (text, graphics and so forth), but the orientation is still that of a document, which is scrolled through in linear fashion.

The direction that Microsoft is taking is that of a data base metaphor. We undertook a study within our own offices to look at the ways people ask about and record data. Our findings showed that the data itself is the key. People generally take a data base approach in recording and accessing information. Someone wanting sales figures for the previous year, for example, would not create a spreadsheet with empty cells and then send it to Accounting to have the cells filled in. Rather, the person would start with the data that he had, and request the additional data needed to complete the picture.

It can be seen that the metaphor question is entirely separate from concepts such as graphics or windowing. It is also a much more difficult issue to deal with. The effort, however, will definitely be worth our while. It is in this area, more than any other, that we can make the breakthroughs that will allow the ordinary user to view the computer as simple. A software approach built around the right metaphor will allow the user to walk up to the machine, immediately see the data that he has put into the system, and then easily choose the application that will allow him to view that data in the format he needs. All this without having to refer to files, spreadsheet cells, formulas, or any other complex constructs.

Issue #4: Tying Personal Computers to Mainframes

A fourth major issue that software developers need to address is the growing interest in tying personal computers into mainframes. Because of the significant differences among mainframes, this is no simple matter. Mainframes (even those made by the same vendor) have different file handlers, different communications software, and different operating systems. The IBM 370 alone has at least six major operating environments, and within each of those, multiple data bases. Creating the software that will allow a personal computer to tie into such a machine will not be a trivial task.

The problem is not simply tying two machines together. We've already done that. We have software that will turn the personal computer into a terminal, ignoring its local intelligence.

The difficulty is to create a method of tying the two together that will allow automatic database query. The user should not, for example, have to know JCL to access data from the mainframe. Nor should he need to learn a complex set of command structures. Rather, he should be able to query data anywhere in the system, and have the system itself use its intelligence to retrieve that data. In fact, the way the data was initially described in the dictionary should tell the system where to go to get it — whether to go, for example, to the mainframe, to CompuServe, or to Dow Jones. Resolving this software issue will not be easy, but it must be accomplished. The increasing use of personal computers in large organizations make this a central concern today.

Issue #5: Expanded Definition of Operating System

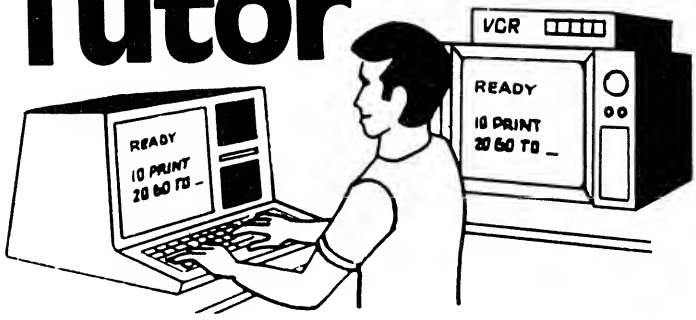
An important development that we will be seeing in the near future is a greatly expanded definition of what an operating system is. Microsoft, for example, as the vendor of one of today's most popular operating systems, MS-DOS, is planning to incorporate an increasingly higher number of functions into that system. Graphics capabilities, user interface capabilities, networking — all will be incorporated into the operating system. Instead of these functions being considered as add-on products, they will be automatically be a part of every machine. This means that applications writers will be able to assume that they are there, and design their packages accordingly.

The Soft World Is Here

As the above observations indicate, a revolution is taking place in the world of computers today, and software is where the innovation is coming from. No longer do we need to go out and build better, more powerful hardware to achieve productivity improvements. We simply develop a new software package and people can put it to use immediately in their existing machines. The revolution is here — and it is soft.

VIDEO INSTRUCTION TAPES

Computer Tutor™



STEP BY STEP INSTRUCTIONS

PICTURES ARE WORTH
THOUSANDS OF WORDS AND SAVE
HOURS OF FRUSTRATION

Use your VCR side by side with your computer to learn disk operating systems, how to program, and how to use programs. Your VCR along with your computer serve as your personal tutor. Pause your VCR to review and learn at your own pace.

VHS or BETA FORMAT

BASIC COMPUTING REVIEW
October 1983 Page 83

"I found the material to be carefully developed and quite appropriate. They covered all the options of the software and even gave some tips that are not in the Profile III Plus manual."

C.O.D. Orders Add $3.00
$3.00 Shipping Per Order
Illinois Add 6% Sales Tax

Specify either VHS or BETA Tape Format

CAT#	TOPIC	APPROX.	PRICE
TAPES FOR THE TRS80 MODEL 1, 3, 4,			
DOS-1	TRSDOS 1.3	90 MIN	**$39.95**
DOS-6	TRSDOS 6.0	105 MIN	**$39.95**
DOS-3	NEWDOS 80	105 MIN	**$39.95**
DOS-2	DOSPLUS 3.5	105 MIN	**$39.95**
DOS-5	LDOS	105 MIN	**$39.95**
DOS-7	CPM 2.2 MONTEZUMA	105 MIN	**$39.95**
DOS-4	MULTIDOS	105 MIN	**$39.95**
EW-1	VISICALC	90 MIN	**$39.95**
EW-7	MULTIPLAN	90 MIN	**$39.95**
WP-1	LAZY WRITER	105 MIN	**$39.95**
WP-2	SUPERSCRIPTSIT	105 MIN	**$39.95**
WP-3	SCRIPTSIT	90 MIN	**$39.95**
WP-4	NEWSCRIPT	105 MIN	**$39.95**
DB-1	PROFILE III PLUS	105 MIN	**$39.95**
DB-2	PFS:FILE	90 MIN	**$39.95**
DIO-3	TRSDOS 1.3 DISK I/O	105 MIN	**$39.95**
DIO-4	TRSDOS 6.0 DISK I/O	105 MIN	**$39.95**
BP-1	LEARNING MOD 3.4 BASIC	180 MIN	**$49.95**
BP-2	LEARNING MOD ONE BASIC	180 MIN	**$49.95**
TC-1	MTERM	45 MIN	**$29.95**
UT-1	DOTWRITER	75 MIN	**$29.95**
TAPES FOR THE TRS 80 COLOR COMPUTER			
BP-5	EXTENDED COLOR BASIC	230 MIN	**$49.95**
TAPES FOR THE COMMODORE 64 AND VIC 20			
BP-3	LEARNING C-64 BASIC	120 MIN	**$49.95**
BP-4	LEARNING VIC-20 BASIC	120 MIN	**$49.95**
DIO-1	COMMODORE 64 DISK I/O	105 MIN	**$49.95**
DIO-2	VIC 20 DISK I/O	105 MIN	**$49.95**
EW-3	CALC-RESULT ADVANCED	90 MIN	**$39.95**
EW-4	CALC-RESULT EAZY	75 MIN	**$29.95**
EW-5	PRATICALC C-64	75 MIN	**$29.95**
EW-6	PRATCALC VIC 20	75 MIN	**$29.95**
EW-9	MULTIPLAN	90 MIN	**$39.95**
WP-5	SCRIPT-64	115 MIN	**$39.95**
UT-2	THE LAST ONE	90 MIN	**$39.95**
TAPES FOR THE IBM PC			
EW-2	VISCALC	90 MIN	**$39.95**
EW-8	MULTIPLAN	90 MIN	**$39.95**
DB-3	PFS:FILE	90 MIN	**$39.95**
TC-2	MTERM	45 MIN	**$29.95**

DEALER INQUIRIES INVITED

LYNN COMPUTER SERVICE

6831 West 157th Street Tinley Park, Illinois 60477 (312) 429-1915

VISICALC IS A TRADEMARK OF VISICORP INC LAZY WRITER IS A TRADEMARK OF ALPHA BIT COMMUNICATION SCRIPSIT, SUPERSCRIPSIT, PROFILE III PLUS AND TRSDOS ARE TRADEMARKS OF TANDY CORP LDOS IS A TRADEMARK OF LOGICAL SYSTEMS INC NEWDOS80 IS A TRADEMARK OF APPARAT INC DOS PLUS AND MTERM ARE TRADEMARKS OF MICRO-SYSTEMS SOFTWARE INC. MULTIDOS IS A TRADEMARK OF COSMOPOLITIAN INC VIC 20 AND COMMODORE 64 ARE TRADEMARKS OF COMMODORE BUSINESS MACHINES, INC. IBM IS A TRADEMARK OF INTERNATIONAL BUSINESS INC CP/M IS A TRADEMARK OF DIGITAL EQUIPMENT CORP NEWSCRIPT IS A TRADEMARK OF PROSOFT. DOTWRITER IS A TRADEMARK OF RCM COMPUTERS TM INDICATES A TRADEMARK OF LYNN COMPUTER SERVICE

#20

Using Model 4 JCL files

Take advantage of the power of TRSDOS 6.0

Model 4

Al Mashburn

So you have this new Model 4 computer that comes with a neat new operating system that does everything but tie your shoe? The only problem is that some of the features are complicated and every time you want to use them, you have to go look it up in the manual so you can enter the commands one-by-one until you get the function you want. An example would be the Memdisk. First you have to enter "system (drive=2,driver="memdisk")". Then questions about which bank, what density, and if you want it formatted are asked. Now, most of the time, the answers are going to be the same, so why should you go through this when you want to use Memdisk, the Comm program, or any other program that requires setup? That's what you got the computer for. Let it do the work.

You may not have realized it, but when you bought your Model 4 disk system, it came with two languages. There is a BASIC interpreter, but there is also a Job Control Language. JCL was designed to run the computer when a pre-determined set of keyboard inputs was needed. Let me say this right now. The JCL that comes with the Model 4 can do a lot more than I'm going to show you here. It is a powerful tool that can literally save hundreds of man (or woman) hours. I am just going to scratch the surface and hopefully whet your appetite. As long as I'm sidetracked, run (don't walk) down to your computer center and get a copy of TRSDOS 6.1 or 6.0.1. The numbers aren't important, but it is necessary that you have the updated DOS. If you bought your machine after September, 1983 you probably already have it. The updates include fixes in the way some JCL jobs are done.

There are three programs I run which require some input to get them set up. The first is BASIC. That is pretty easy to remember, but I always forget to specify one file for maximum memory. I go into BASIC, then remember that I forgot (F=1) again, exit BASIC and reenter the right way. As I mentioned, Memdisk is another program that goes through setup as does Comm. So, I wrapped up all three into what I call SHELL/JCL. A shell is a program that sort of insulates you from the operating system. It is really nice if someone who is a little computer shy has to use the machine.

There are no linenumbers in JCL, so it's a little hard to follow the program flow in large jobs. It might be a good idea to study the owner's manual if one of the terms or commands seems confusing.

The first line of Shell is a comment line. It just so happens that we want a comment line at the start of the job to list the options, but even if we didn't need a comment, all JCL programs should start with a comment because some won't even run without it. All comment lines start with a period.

The second line uses JCL command Keyin. Keyin allows input of a number from 0 to 9 and works like the BASIC INKEY function. That is, you don't have to press enter after selecting a number. The slashes (//) before Keyin are the way JCL tells between its own commands and those it is issuing to the computer. Think of Keyin as

Listing 1 — Shell/JCL

```
.(1) BASIC        (2)   MEMDISK       (3)  COMM    (4) DOS
//KEYIN   YOUR CHOICE
//1
BASIC (F=1)
//STOP
//2
SYSTEM (DRIVE=2,DRIVER="MEMDISK")
D
D
Y
//EXIT
//3
SET *CL COM/DVR  (B=300,W=8,S=1,P=OFF)
COMM *CL
//STOP
///
.EXITING TO DOS
//EXIT
```

BASIC's "On X GOTO." When you key in a choice, the program searches through the job by looking at the //# blocks. In other words, if you were to enter 3, it would search past //1 and //2 until it got to //3. Then it would execute the commands after //3. In this case, it would issue the commands necessary to go into the Comm program. You will note that there are no leading periods or slashes in the commands after the //#. When the program comes to one of these lines it is just like you typed them in from the keyboard.

At the end of each block (//1, //2, etc.) there is another JCL command. In Shell it is either //EXIT or //STOP. These tell the computer what to do when it has done the job. //STOP means to stop the job and hand control over to the program you are in. In other words, if you are in BASIC when the job is done, you will be left in BASIC. Sometimes this isn't good. In the Memdisk job, if you //STOP at the end, you are left hanging with no TRSDOS Ready or anything, so we //EXIT. //EXIT returns us to the TRSDOS Ready prompt no matter where we are when the job is done. Do not use //EXIT unless that's where you want to be. If you use it in a job like block //3, you would enter the Comm program and then exit back to TRSDOS. That's like building a boat in your house then tearing it apart to get it outside. You're right back where you started.

Near the end of the program you'll see three slashes (///); these mean "there are no more blocks past here." If you were to enter 8 at //Keyin, the program would search past block //3 and come to ///. The program would stop looking for block //8 and would execute the command following /// (in this case, //EXIT to TRSDOS). The D,D,Y in block 2 are the answers you must give to set up Memdisk for a 128K machine. If you have a 64K, you will have to change it.

If you do have a job that you would like to do with this program, or you want to change some of the commands in this one, you will need some way to enter the JCL program (it is not entered from BASIC). The operating system comes with the Build command to enter JCL programs. The syntax to enter this program would be BUILD SHELL/JCL. Build has no editing features so if you made a mistake (some of us do) you would have to "Remove" the program and retype it. When you Build a program, there cannot be a program with that name on the disk. The best way to make small changes is to use a word processor that can save text in ASCII form. In Scripsit, you would SAVE SHELL/JCL,A. To run this program, enter it as shown using Build. Then enter DO=SHELL/JCL. The reason for the = is that there is no compiling needed so the = tells the computer to execute without compiling. Once you are satisfied that the job works as you want it to, go to TRSDOS ready and type AUTO DO=SHELL/JCL. Then press reset and when the computer boots it will run Shell. This will save you a lot of time so you can go out and make a lot of money and buy more computer equipment.

The affordable step up in home computing.

Radio Shack's Best Is Now

Expand Your Programming Power! Our new 64K Extended BASIC Color Computer 2 gives superb features to serious programmers—at a very low price. It attaches easily to any television, and has an electric typewriter-quality keyboard in a compact, white case. Using the built-in Extended BASIC language, you can write sophisticated programs and easily create high-resolution graphics. Produce drawings, charts, diagrams and animation with simple, one-line commands. "Draw" with eight different colors, create musical tones, analyze data and much more. Or select from our large library of instant-loading Program Pak™ cartridges for games, education and budgeting.

Deluxe Joystick

39.95 Each
26-3012

TV not included

64K Extended BASIC Color Computer 2

AS LOW AS $26 PER MONTH

Radio Shack CITILINE

259.95
26-3127

Prices apply at participating Radio Shack stores and dealers. OS-9 is a trademark of Microware and Motorola, Inc.

TRS-80® Color Computer Even Better.

The Heart of a Professional Disk-Based System. Add a Color Computer disk drive for an extra 156,000 characters of storage and quicker access to your data. You can add up to four disk drives for a vast 626,000 characters of total storage. For advanced programmers, our new OS-9 operating system (26-3030, $69.95) allows you to utilize the full 64,000-character memory for assembly language programs. OS-9 includes a text editor, assembler and debugger. You can develop and edit your own assembly language programs, convert them into machine language for execution, then test them for errors.

Easily Expandable. Best of all, your 64K Color Computer 2 will grow with your future needs. Add our Deluxe Joysticks for more accurate cursor control and faster response with your favorite games. Or choose our Multi-Pak Interface, which lets you connect up to four Program Paks to your Color Computer at once. When you're ready to change programs, just move the selector switch. Go on-line with the world with our deluxe RS-232 Program Pak (26-2226, $79.95). You can communicate with national information services and local bulletin boards by telephone—just add a modem. You can also add a printer, a Color Mouse cursor controller and more!

See It Today. The 64K Extended BASIC Color Computer 2 is available today at over 1000 Radio Shack Computer Centers and at participating Radio Shack stores and dealers nationwide.

Multi-Pak Interface

179.95

26-3024

USE YOUR Radio Shack CITILINE

Disk Drive #0 Kit

399.95

26-3029

AS LOW AS $28 PER MONTH

Radio Shack CITILINE

Send me the all-new 1984 TRS-80 catalog RSC-11.

Mail To: Radio Shack, Dept. 84-A-596
300 One Tandy Center, Fort Worth, Texas 76102

NAME _____

COMPANY _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

TELEPHONE _____

Radio Shack®

The biggest name in little computers®

A DIVISION OF TANDY CORPORATION

Computer ease

Making programs transportable

For all readers

Mark E. Renne

Both Frank Hatchett of Bowling Green, Kentucky, and Ralph Martin of Phoenix, Arizona, had the same question: "Will programs written in BASIC for one computer work on another computer that uses BASIC?" This month's column answers this age-old question and includes some ideas to make programs easier to convert from one machine to another. Programmers, are you listening?

So, what's the answer, yes or no? It turns out the answer is maybe. If the person that wrote the program was a caring, loving, computer scientist whose life revolves around writing code that works on every machine made, it probably will be easy to convert. If the program was written to work on a Model 47 with a high-rise hemi-clock using BOBDOS 4.1.3.4, you'll probably never be able to convert it for use on your Model III.

What's transportable? Transportable means that the program is designed to work with a number of different machines with little or no conversion. This is just what we're looking for! Our friend who has one brand of computer can write programs that we can use on our computer. The programs, to be truly transportable, are written for a particular operating system and/or a particular version of a language. Of course, both parties must have this particular operating system and/or language to make the program transportable.

So, why won't BASIC work? The biggest problem is that there is no such thing as a standard version of BASIC for microcomputers. Sure, ANSI (American National Standards Institute, the guys who make up standards) has a standard for BASIC, but it is so limited in function that no manufacturer would use it. This means that every computer maker can invent a BASIC full of strange, unusual commands to confuse computer users worldwide. Hence, there is no such thing as "standard" BASIC.

What if we all have Microsoft BASIC? My apologies to Bellevue, but Microsoft has done more to confuse the issue than all the other software companies combined. There is no such thing as "standard" Microsoft BASIC. Even though a computer has Microsoft BASIC, the computer may or may not run programs written in Microsoft BASIC on your computer. How can that be? Since Microsoft has to keep up with all the other companies making BASIC, they continually add new features to take advantage of a particular computer. Sometimes they add unique features at the request of the hardware manufacturer. That turns out to be terrific if you own that new computer, but very troublesome if you own an older computer.

Isn't there any hope at all? Absolutely! In fact, I have some guidelines to offer that will make programs very easy to convert. It turns out that all BASICs have a great number of commands in common and only a few that differ. If we all stick to the commands that we have in common, we'll have no problems at all. Of course, we have to agree what those commands are.

When will a program work on my computer? If (and only if) the program is command-, screen-, processor-, operating system-, and format-compatible will it positively work on your computer 100 percent of the time without modification. In other words, a Model III program will run on another Model III. If any item is different, the program will probably require modification on your part to run on your computer.

What's command compatibility? All of the commands used in the program must be common to both BASICs involved. For example, most (but not all) Radio Shack computers use the command "CLS" to clear the screen and home the cursor. The Apple computer uses the command HOME. To use a program written for the Apple, you'd have to change all the HOMEs to CLSs. This type of command incompatibility is not unique to different brands of computers as there are several command differences between Radio Shack models. For example, the command DEFDBL, which defines a variable as double precision, is found on most Radio Shack computers but not the Color Computer. Of course, the Color

Computer has many commands that deal with color and graphics not found on any of the other computers.

What about screen compatibility? Most of the Radio Shack computers (Models II/4/12/16/2000) use an 80 by 24 screen. That would be 80 characters across by 24 lines. The Model I and III have 64 by 16 screens and the Color Computer has only 32 characters across. This means that the larger screens can hold nearly twice as many characters as the smaller screens. This would lead to problems with programs that expect the screen to be of a certain format. A "PRINT @ 120, X" would print the contents of X in very different places on the screen depending on the computer it was used on.

If the program expects high-resolution graphics capabilities, you'll have trouble converting it to a non-graphics computer. Any programs that use SOUND or COLOR commands must be used on the Color Computer or Model 2000. There are no similar commands on the other computers. This also applies to any special characters like those found on the Model III. There's no way to convert the "little spaceship" found on the Model III to a spaceship on the Model II. Imagine building a business computer that can't display a spaceship!

What's processor compatibility? This means that both computers have the same Central Processing Unit (CPU). This isn't very important when writing in BASIC unless the programmer has used embedded machine code. Machine language commands on a Z-80 and 6502 are quite a bit different. Programmers use machine code in BASIC programs for a variety of reasons. It speeds up graphics and sorting. It can also be used to produce sound effects by using the cassette port. Although machine code can make a program more efficient, it also can make a program impossible to transport.

What's format compatibility? It's a sad fact that the only way to get a program written for one computer into another computer is by typing the program in from the keyboard. Each computer writes the program out to tape or disk using a unique format. This means that a disk for one computer can't be used on another computer. (Some computers have convert programs to overcome this problem. —Ed.) I guess this means there is no such thing as format compatibility. Actually, there is another way to get a program from one computer to another: use the serial port. We'll cover this and other telecommunication features in another column.

How do we make programs more transportable? Only use commands that are common to all computers. There are several books that list or cross-reference different BASICs. If you must use an uncommon command, comment exactly what that piece of code does. Where possible, use variables to represent specific values instead of numbers. This would allow you to change all occurrences of this value with one change instead of searching for all the places to change.

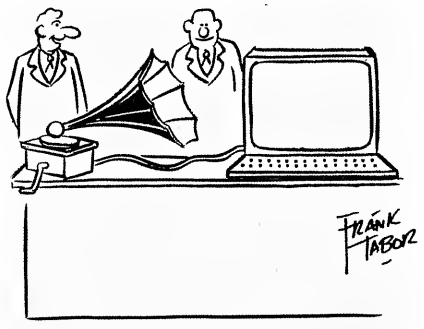
Don't use machine code within a program. If you must, write comments to indicate exactly what the machine code does. If you used "POKE 16412,1" in a program, indicate that this simply keeps the cursor from blinking. Perhaps the computer we're converting the program to has a command to duplicate this function. It might even be that we don't need to have this function to have an effective program.

Break up the program into small parts or modules. This is part of what is called structured programming or top-down design. Indicate what each part does with complete comments. There might be several ways to write the same code, but you have to know what it does before you can convert it.

Keep the program simple! Sure, it's nice to PEEK and POKE a little machine code here and there, but it makes your programs useless on other computers. By keeping the function of each line within the "universal" BASIC commands, you can make program conversion a breeze.

I know that we can never expect commercial software houses to produce BASIC code using these guidelines, but you can write your programs using them. *Basic Computing* has always believed in BASIC and the fact that it is the one thing all Radio Shack computers have in common. Will one program, written on one computer, run on another? If you follow the right method, the answer is yes.

Thanks to everyone who's written. If you have a question, feel free to write in care of *Basic Computing*, 3838 S. Warner Street, Tacoma WA 98409. I can guarantee a fast, friendly reply, even if I don't have all the answers. Next time, telecommunications and you. Computers are fun and understanding them is easier than you think.



"Interesting add-on."

Golf league recordkeeping

Let the Color Computer do the work

Color Computer

Lynn Davis

I have played in golf leagues for many years. One of the things about them that I have noticed is that the league secretary doesn't seem to hold that position for a very long period of time.

It's not hard to understand, however, why this person suffers from "golf league secretary burnout." When you consider that he or she is responsible for 20 to 50 golfers, calculating all the handicaps, keeping track of all the points won and lost, ranking teams and golfers in order, and putting out an accurate weekly schedule that includes all this material; it becomes easy to see why a normal human being would not want this position for any extended period of time.

In our case, the position of golf league secretary gets shuffled around quite a bit. If my experience is any indication, you can expect to be asked to do it a lot sooner than you think. As soon as someone in the league finds out you own (or have access to) a computer, you will become a prime candidate.

You can also expect not to find any commercially-available golf league programs. No two golf leagues work exactly the same way. Some have two-man teams, others have more. Some use a nine-point system, others don't. Very few follow USGA rules when calculating handicaps.

The league that I play in is just as different as all the rest. I believe, however, that the program I've designed will give you a good start if you have to devise one of your own.

League Framework

I play in a nine-hole league. We have a total of eight teams, with six golfers on a team. Each golfer has a position on a team. The golfer with the lowest handicap has position number one, the highest handicapped golfer occupies the number six position.

Two golfers from different teams (but having the same position) play each other for a total of eleven points. Handicaps are based on 96% of the difference between the golfer's gross score and par.

In our golf league, we must also keep track of points won by a substitute, a split season, and the changing positions of golfers as their handicaps change.

The most important function is the arranging of all this information into an accurate weekly schedule. Not only do we need a printed schedule that shows who plays whom, but it should also tell what the individual handicaps are, how many points golfers have won, how many points substitutes have won, total team points and teams ranked by total points.

Finally, our league also has a weekly competition where we find out which golfers with the same position shot the lowest score with handicap. Winners in each position get a free golf ball.

The Program

With the above parameters determined, I wrote the program shown. Line 160 CLEARS the string space needed for the storage of team names and golfer names.

Line 170 establishes the two arrays needed to store the league information. The array labeled "TA" contains all the statistics for every team and every golfer. The array "NA$" stores the team names and golfer names.

The 56 rows in both arrays result because there are 48 golfers plus 8 teams to account for in the league. The 12 columns in the "TA" array are used to store the numbers input or calculated by the computer. The following list summarizes what is found in each column if it contains information for a golfer.

Column	Information
1	Current playing position.
2	Starting handicap.
3	Handicap for the current week.
4	Handicap from previous week.
5	Handicap from three weeks ago.
6	Handicap from four weeks ago.
7	Handicap from five weeks ago.
8	Total points won.
9	Gross score for current week.
10	Net score for current week.
11	First hole score for current week.
12	Current playing handicap.

If the row is used to store numbers for a team, the columns contain the following information:

Column	Information
1	Team number.
3	Team points won during the second half.
4	Flag used when sorting.
6	Total points won by the team.
7	Team points won during the first half
8	Points won by substitutes

(The other columns are not used.)

Figure 1 shows a printout of the information contained in both the "NA$" and "TA" arrays at the end of the 16th week of play.

Initial Menu

Lines 230 to 290 are the initial menu. If this is the first computer run for the season, the program GOSUBs to lines 3840 to 3980. In this section, the starting information for each team, plus the six golfers per team, are entered.

If this is not the first week, the program GOSUBs to lines 2610 to 2810. This section will load the appropriate information from a cassette file into the proper array. The file contains the information stored in the "TA" and "NA$" arrays.

The Main Operating Menu

Lines 310 to 430 are the main

operating menu. From this point in the program, I can do any of the following: Set all flags to zero, check or change figures, save the information on tape, do the weekly update, end the program, or advance to the print menu.

Setting flags to zero: There are two types of flags that must be cleared (or set to zero) in order for the program to operate correctly. Both flags must be set before the weekly input of new information is started.

The first flag set to zero is the first-hole score for all golfers. These scores, or the lack of them, are used by the computer to determine whether the golfer played during the current week.

If he did not play, the zero will tell the computer to skip the golfer (in line 1000) when looking for the weekly golf ball winners.

The second flag is used when the program sorts the teams and their total scores in order. Once a team has been matched with its point total, this flag is set in line 1760. A flag is needed in case there are ties. It stops the computer from assigning a team name to more than one point total.

Check/change figures: If this option is selected, the program GOSUBs to line 2200 and allows you to check or change any names or numbers in the array. This is essential because of the inevitable input error, or for the input of a new golfer that takes the place of someone who is leaving.

The weekly league update: If this selection is chosen from the main menu, the program GOSUBs to line 450. After inputting the number of weeks of league play (needed to determine if the points are for the first or second half of the season), you are ready to input the weekly information from the scorecards.

The first information entered is the name of the golfer, or the team name if the input is for a substitute. Lines 550 to 600 find the position of the appropriate golfer of team in the array.

Once the program knows which golfer or team the information is being entered for, line 610 asks for the number of points won. If the points are for a team, which is determined in line 630, the program then goes to line 690 and asks if the information is correct.

If the information to be input at line 610 is for a golfer, the program asks where the competition took place. This question is important only if the golf course has a different par for the front and back.

Lines 660 to 680 ask for input of the gross score, net score, and the score on the first hole. The gross score is needed to calculate handicaps and the net score and first hole score are needed to determine who the golf ball winners are.

Line 760 moves the previous handicaps stored in the array and deletes the oldest handicap if more than five weeks of play have occurred. Line 780 or 790 calculates

the current handicap (depending on which side of the golf course the competition took place).

Line 810 is included for those few golfers who shoot a perfect round of golf and, therefore, have a zero handicap for the week. Because lines 830 to 850 use a lack of scores (or zeroes) in the handicap section of the array to figure how many weeks of golf have been played, I had the computer assign a zero handicap the value of 0.0001. This number is small enough to not affect the golfer's real handicap, but is large enough to show the computer that a zero is not stored there.

Lines 830 to 920 are used to calculate the golfer's playing handicap. The *playing* handicap is determined by two factors: (1) how many weeks of golf have been played and (2) the *weekly* handicaps stored in the "TA" array.

Line 930 sends the program back to the update section for input of information concerning the next golfer.

In the update section, you can also return to the main menu at any time by entering "MM." From here, you can check/change figures, or even save a partially-completed update. Being able to save a partially-completed update is made possible because the flags are manually set only when needed.

Once the update is finished, and "UF" has been entered, the program will GOTO line 950. Lines 950 to 1050 will print on the screen the names of all the golfers in each flight (i.e., those with the same position number) that played golf in that particular match. Also printed will be the net score and the first hole score for each golfer.

With this information, I can decide who the golfer is that shot the lowest score with handicap, and thus find the winners of the weekly golf balls. The first-hole score is used only to break ties, and is similar to a sudden-death playoff. The names of the ball winners and their net scores are stored for later printout in two small arrays in line 1040.

Lines 1130 to 1400 are used when it is time for a position change. Each member of a team is printed on the screen with his handicap and his current playing position. If a golfer's handicap has changed enough so that he is better (or worse) than other golfers on his team, this section of the program allows me to rearrange the position numbers stored in the array.

Lines 1440 to 1790 add up the team's total points, sorts them in decreasing numerical order, and matches up the team name with the appropriate point total.

The total points, arranged in order, are stored in an array labeled LL() in line 1660. The team names, in order, are stored in an array labeled H$(F) in line 1760. Lines 1800 to 1870 print the teams and point totals, in order, on the computer screen.

The print menu: Lines 1800 to 1990 contain the print menu, along with a few other functions that might be useful at this point of the program.

Printout of all figures: If this option is selected, the program will GOSUB to lines 2830 to 2930 and print out all the golfer and team names plus all the corresponding numerical figures stored in the two large arrays.

For the printout of a weekly playing schedule, the program GOSUBs to lines 3080 to 3820. A typical printout is shown in Figure 2.

Line 3230 asks for the input of the numbers of two teams that are to play each other. The computer locates these teams, sorts the golfers by their playing positions, and temporarily stores the information

TEAM INFO OR GOLFER	TM# POS	--- SHP	2HP HC1	FLG HC2	--- HC3	TPT HC4	1HP HC5	SPT PTS	--- GRS	--- NET	--- HL1	PHC
HEID'S	11	0	285.0	1.0	0.0	539.5	254.5	63.0	0	0	0	0.00000
BOHART	1	4	2.9	3.8	4.8	2.9	9.6	72.0	40	36	5	3.84000
HARRIS	2	5	4.8	6.7	1.0	7.7	5.8	71.0	42	35	4	5.76000
SARGENT	4	9	8.6	7.7	6.7	3.8	8.6	86.5	46	38	6	7.68000
DAVIS	3	10	7.7	6.7	5.8	6.7	6.7	94.0	45	38	6	6.72000
RANDALL	5	12	9.6	7.7	13.4	17.3	7.7	87.5	47	36	4	10.24000
HOGAN	6	14	15.4	18.2	19.2	15.4	9.6	65.5	53	37	4	16.32000
GULF	22	0	275.5	1.0	0.0	547.5	272.0	36.0	0	0	0	0.00000
CARNEY	3	5	6.7	8.6	11.5	8.6	9.6	79.5	44	35	4	8.96000
SCHIRATO	1	3	4.8	2.9	3.8	1.0	2.9	84.0	42	39	5	3.20000
HOWARD	2	8	5.8	7.7	7.7	5.8	7.7	81.0	43	36	6	7.04000
CAPELLA	5	11	13.4	9.6	12.5	12.5	10.6	92.0	51	39	4	11.84000
THOMAS	4	12	6.7	10.6	9.6	11.5	9.6	85.5	44	34	4	9.92000
HATTER	6	17	14.4	10.6	14.4	10.6	23.0	89.5	52	39	8	13.12000
FLAMINGO	33	0	270.5	1.0	0.0	542.5	272.0	70.5	0	0	0	0.00000
SANTAFERRA	1	4	2.9	4.8	2.9	8.6	2.9	67.0	40	36	5	3.52000
HIRONS	3	5	6.7	6.7	10.6	8.6	0.0	91.5	44	37	4	7.36000
PAUL	2	8	6.7	7.7	3.8	4.8	6.7	91.5	44	38	6	6.08000
BROWNING	6	11	8.6	13.4	9.6	14.4	16.3	64.5	46	33	6	12.48000
RICHBERG	4	0	7.7	11.5	13.4	11.5	10.6	78.5	45	33	5	11.20000
SNYDER	5	0	15.4	6.7	16.3	4.8	12.5	79.0	53	42	7	11.52000
MOORE	44	0	257.0	1.0	0.0	515.5	258.5	51.5	0	0	0	0.00000
TILDEN	2	4	5.8	1.9	3.8	4.8	4.8	87.0	43	39	6	4.48000
RUSH	1	6	8.6	2.9	1.0	7.7	1.9	45.5	46	44	4	4.16000
REISSIG	3	0	5.8	8.6	5.8	8.6	5.8	96.5	43	36	5	6.72000
WEBSTER	4	10	6.7	11.5	4.8	6.7	9.6	78.0	44	37	6	7.68000
LAWRENCE	5	13	13.4	12.5	9.6	13.4	13.4	83.5	51	38	5	13.12000
MASTERPOLE	6	16	17.3	12.5	14.4	14.4	15.4	73.5	55	41	8	14.72000
McARDELL	55	0	250.0	1.0	0.0	506.0	256.0	68.5	0	0	0	0.00000
MORRISSEY	1	5	0.0	4.8	1.0	1.9	3.8	88.5	37	33	4	2.24000
WERTHMAN	2	7	4.8	6.7	7.7	2.9	8.6	66.5	42	34	4	6.40000
KRAUSHAUR	5	9	9.6	13.4	19.2	13.4	11.5	48.5	47	32	5	12.80000
FOX	3	11	8.6	10.6	15.4	10.6	11.5	73.5	46	34	4	10.88000
JANTSCH	4	13	14.4	13.4	8.6	10.6	9.6	88.5	52	41	5	11.20000
KING	6	16	13.4	15.4	15.4	9.6	17.3	72.0	51	35	5	14.72000
HARDTER	66	0	238.5	1.0	0.0	508.0	269.5	17.0	0	0	0	0.00000
MURPHY	1	5	3.8	4.8	2.9	4.8	8.6	69.5	41	36	6	4.48000
PECK	2	5	5.8	7.7	6.7	7.7	4.8	75.5	43	36	5	6.72000
SHIFLET	3	8	13.4	8.6	6.7	3.8	10.6	84.5	51	43	6	8.64000
O'LEARY	5	11	6.7	11.5	18.2	12.5	13.4	80.5	44	31	8	12.48000
TOSCANO	4	12	13.4	11.5	7.7	5.8	12.5	96.5	51	41	5	10.56000
TRAUB	6	14	14.4	16.3	9.6	10.6	17.3	84.5	52	39	6	13.76000
FUTURAMA	77	0	270.5	1.0	0.0	547.5	277.0	23.5	0	0	0	0.00000
SCHMITT	1	3	6.7	-1.9	-1.0	1.0	1.9	100.5	44	44	5	0.64000
BIASI	3	7	8.6	8.6	4.8	6.7	9.6	87.5	44	36	8	8.00000
ROSSO	6	8	10.6	16.3	11.5	13.4	6.7	89.5	48	36	7	11.84000
MEAD	5	11	9.6	16.3	7.7	10.6	11.5	83.0	47	35	5	10.56000
RUDY	4	12	9.6	9.6	10.6	6.7	10.6	72.5	47	37	5	9.92000
MICK	2	0	5.8	7.7	3.8	4.8	3.8	91.0	43	38	6	4.80000
MAURER F. H.	88	0	265.0	1.0	0.0	517.5	252.5	47.5	0	0	0	0.00000
MAURER	3	5	8.6	3.8	6.7	8.6	5.8	69.5	44	39	0	7.04000
GRIMSLEY	1	6	2.9	4.8	6.7	4.8	10.6	83.5	40	35	6	5.44000
MOYER	2	8	5.8	10.6	5.8	8.6	5.8	95.5	43	36	6	6.72000
SNELL	4	10	14.4	7.7	13.4	11.5	7.7	79.5	52	42	6	10.88000
HIEPLER	5	14	16.3	10.6	12.5	21.1	8.6	62.5	54	41	7	13.12000
BAILEY	6	14	13.4	15.4	14.4	17.3	14.4	79.5	51	35	5	14.72000

Figure 1 — This is a printout showing all the information stored in the name array (NA$) and in the team or golfer (TA) array. To tell if the name is for a team or individual golfer, look at the first number to the right of the name. If the number is 11 or larger, the name is for a team.

in two small arrays (lines 3390 and 3500). Once this is done, the computer prints out the corresponding section of the schedule. The program repeats this section three more times, so that all eight teams are accounted for.

Listing 1

```
10 REM **************************
20 REM
30 REM    GOLF LEAGUE PROGRAM
40 REM
50 REM BY:
    MR. LYNN H. DAVIS
60 REM      4316 AMBLEWOOD LANE
70 REM      CLAY, NEW YORK 13041
80 REM
90 REM   TRS-80 COLOR COMPUTER
100 REM  32K EXTENDED BASIC
110 REM  CASSETTE TAPE RECORDER
120 REM  LINE PRINTER VII
130 REM
140 REM ************************
150 CLS
160 CLEAR 850
170 DIM TA(56,12):
    DIM NA$(56)
180 PRINT@196,"LIVERPOOL BUSINESSMAN'S"
    ;
190 PRINT@262,"GOLF LEAGUE PROGRAM";
200 GOSUB 4000
210 FOR TT=1 TO 1500:
    NEXT TT
220 CLS
230 REM INITIAL MENU TO INPUT GOLFERS A
    ND TEAMS OR TO LOAD THEM FROM
     TAPE
240 PRINT"INITIAL MENU"
250 PRINT:
    PRINT"1. STARTING NEW"
260 PRINT"2. LOAD FROM TAPE"
270 PRINT:
    INPUT"<ENTER> NUMBER";S
280 IF S<1 OR S>2 THEN 220
290 ON S GOSUB 3840,2610
300 GT=0
310 CLS
320 REM THE MAIN OPERATING MENU
330 PRINT"MAIN MENU"
```

LIVERPOOL BUSINESSMAN'S GOLF LEAGUE — MATCH NUMBER 16 — FRONT SIDE

TEAM / NAME	HCP	POINTS	VS	TEAM / NAME	HCP	POINTS
HEID'S		63.0		MAURER F. H.		47.5
1. BOHART	4	72.0		1. GRIMSLEY	5	83.5
2. HARRIS	6	71.0		2. MOYER	7	95.5
3. DAVIS	7	94.0		3. MAURER	7	69.5
4. SARGENT	8	86.5		4. SNELL	11	79.5
5. RANDALL	10	87.5		5. HIEPLER	13	62.5
6. HOGAN	16	65.5		6. BAILEY	15	79.5
2nd HALF TOTAL		285.0		2nd HALF TOTAL		265.0

TEAM / NAME	HCP	POINTS	VS	TEAM / NAME	HCP	POINTS
GULF		36.0		FUTURAMA		23.5
1. SCHIRATO	3	84.0		1. SCHMITT	1	100.5
2. HOWARD	7	81.0		2. MICK	5	91.0
3. CARNEY	9	79.5		3. BIASI	8	87.5
4. THOMAS	10	85.5		4. RUDY	10	72.5
5. CAPELLA	12	92.0		5. MEAD	11	83.0
6. HATTER	13	89.5		6. ROSSO	12	89.5
2nd HALF TOTAL		275.5		2nd HALF TOTAL		270.5

TEAM / NAME	HCP	POINTS	VS	TEAM / NAME	HCP	POINTS
FLAMINGO		70.5		HARDTER		17.0
1. SANTAFERRA	4	67.0		1. MURPHY	4	69.5
2. PAUL	6	91.5		2. PECK	7	75.5
3. HIRONS	7	91.5		3. SHIFLET	9	84.5
4. RICHBERG	11	78.5		4. TOSCANO	11	96.5
5. SNYDER	12	79.0		5. O'LEARY	12	80.5
6. BROWNING	12	64.5		6. TRAUB	14	84.5
2nd HALF TOTAL		270.5		2nd HALF TOTAL		238.5

TEAM / NAME	HCP	POINTS	VS	TEAM / NAME	HCP	POINTS
MOORE		51.5		McARDELL		68.5
1. RUSH	4	45.5		1. MORRISSEY	2	88.5
2. TILDEN	4	87.0		2. WERTHMAN	6	66.5
3. REISSIG	7	96.5		3. FOX	11	73.5
4. WEBSTER	8	78.0		4. JANTSCH	11	88.5
5. LAWRENCE	13	83.5		5. KRAUSHAUR	13	48.5
6. MASTERPOLE	15	73.5		6. KING	15	72.0
2nd HALF TOTAL		257.0		2nd HALF TOTAL		250.0

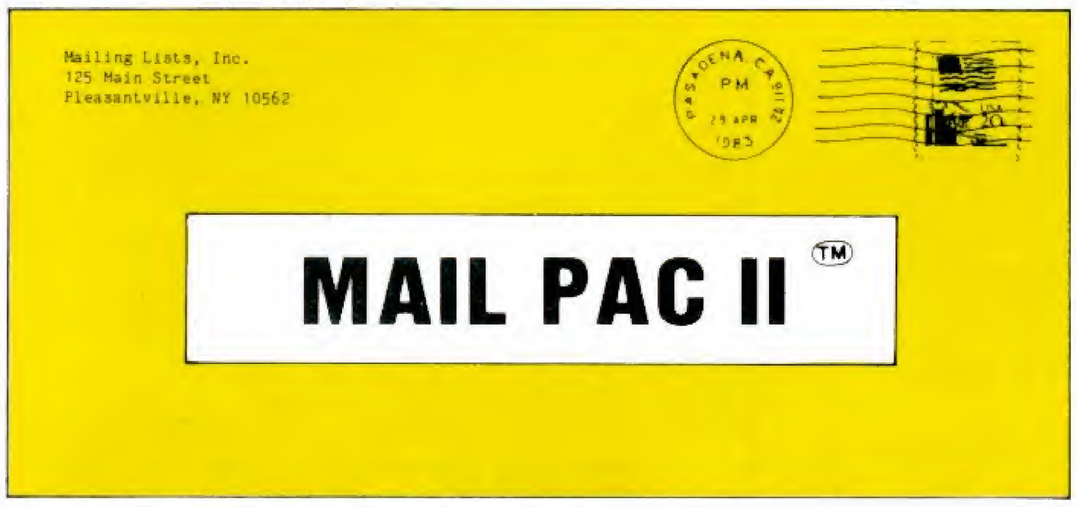
BALL WINNERS	SCORE
1. MORRISSEY	33
2. WERTHMAN	34
3. FOX	34
4. RICHBERG	33
5. O'LEARY	31
6. BROWNING	33

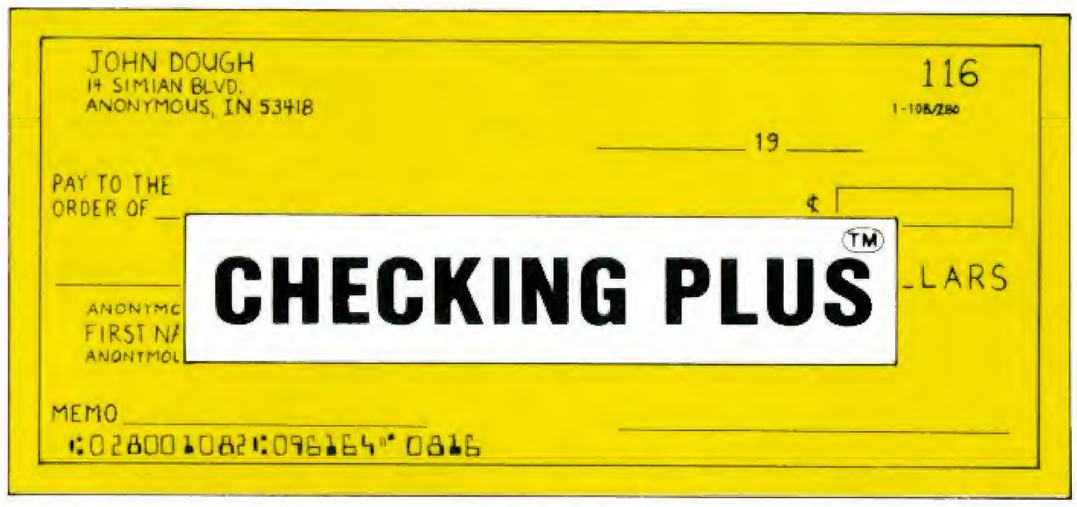
TEAM STANDINGS	POINTS
1. HEID'S	285.0
2. GULF	275.5
3. FLAMINGO	270.5
4. FUTURAMA	270.5
5. MAURER F. H.	265.0
6. MOORE	257.0
7. McARDELL	250.0
8. HARDTER	238.5

Figure 2 — This is a printout showing a typical weekly schedule. All of the information in this printout can change weekly, depending on the golfer's handicap, number of points won, whether his position will change, and which teams will be playing against each other.

ANYONE CAN USE OUR SOFTWARE!

MAIL PAC II and CHECKING PLUS represent a new generation of computer software. Our software doesn't even need a manual! Anyone can quickly learn to use either package by following simple, explicit on-screen instructions. But, for those of you who insist -- we've included a complete user's manual as well.





MAIL PAC II FEATURES:

High Capacity --

Stores from one to one million names in Zip Code, Numerical or Alphabetical order. The only limitation is your disk storage space.

Flexibility --

Prints your mailing list on 1, 2, 3, or 4-across labels (with up to 5 user-defined lines on each label) or as a compact, user-designed directory. The record length is completely user-defined, and each field within each of your records is completely variable (allowing storage of any number of characters for any particular address entry). Devote fields to telephone numbers, codes, or even special messages related to each particular name on file.

Powerful --

A built-in word processor allows you to create personalized form letters for each address on your list (or just a particular group of addresses).

Ease of Use --

Create new mailing lists, review existing lists, handle changes of address, delete cancelled names, sort lists, and purge duplicate names from your files. Complete on-screen instructions tell you in plain english exactly how to accomplish all of these tasks.

MAIL PAC II $99.95

Both Programs Require 48K And At Least One Disk Drive (2 Drives For Higher Storage Capacities). And All Hard Disk Systems Are Supported.

CHECKING PLUS FEATURES:

Complete Check Register --

Checking Plus stores your entire check register in a disk file, and then uses the data to balance your account, track your expenses, and help you make budget projections. Review the entire checkbook, enter checks, deposits, fees and adjustments, mark outstanding checks when paid, and get an instant cash balance at any time. All data can be viewed onscreen or printed out in report form.

Tax Preparation --

Storage of monthly and yearly totals and other important information aids in income tax preparation, for your personal use or for your accountant.

Handles the Details --

Store names and addresses for frequently written checks, then print checks to fit standard window envelopes, eliminating the need for extra typing. You can even store any comment, explanation or other message (up to 255 characters) related to a particular check.

Automatic Monthly Bill Payments --

Enter amounts and names of payees for all of your fixed-cost monthly payments, and then sit back while the system automatically prints checks.

CHECKING PLUS $99.95

FREE--TRS-80 Mod I,III & 4 programs supplied on DosPlus (minimum system). Complete DosPlus also available.

FOR YOUR TRS-80™ • APPLE™ • IBM PC™ • NEC™ • OSBORNE™ • XEROX™ • KAYPRO™ • TELEVIDEO™ • ZENITH™ • SANYO™
DEC™ • TI PROFESSIONAL COMPUTER™ • SUPERBRAIN JR.™ • EPSON™ • Any CP/M™ Computer

CP/M-based Computers must be equipped with Microsoft BASIC (MBASIC or BASIC-80)

OUR SOFTWARE CATALOG

H & E Computronics, Inc., has mailed more than 1 million software catalogs since 1978. Send $2 for our new 64-page catalog today! (We also send you our catalog FREE with every order). DEALER INQUIRIES WELCOME

30-DAY MONEY BACK GUARANTEE

*** ALL PRICES & SPECIFICATIONS SUBJECT TO CHANGE ***
DELIVERY SUBJECT TO AVAILABILITY

H&E COMPUTRONICS INC
50 N. PASCACK RD., SPRING VALLEY, N.Y. 10977

24 HOUR ORDER LINE
(914) 425-1535

ADD $3.00 FOR SHIPPING IN UPS AREAS
ADD $4.00 FOR C.O.D. OR NON-UPS AREAS
ADD $5.00 TO CANADA & MEXICO
ADD PROPER POSTAGE OUTSIDE OF U.S., CANADA & MEXICO

NEW TOLL-FREE ORDER LINE
(OUTSIDE OF N.Y. STATE)
(800) 431-2818

```
340 PRINT:
    PRINT"1. set all flags to zer
    o"
350 PRINT"2. CHECK/CHANGE FIGURES"
360 PRINT"3. SAVE ON TAPE"
370 PRINT"4. WEEKLY UPDATE"
380 PRINT"5. END PROGRAM"
390 PRINT"6. PRINT MENU"
400 PRINT:
    INPUT"<ENTER> NUMBER";X
410 IF X=6 THEN GOTO 1890
420 ON X GOSUB 2950,2200,2010,450,4170
430 GOTO 310
440 REM WEEKLY LEAGUE UPDATE
450 CLS:
    PRINT"HOW MANY WEEKS OF GOLF
    HAVE"
460 INPUT"BEEN PLAYED";WN
470 CLS
480 PRINT"       WEEKLY LEAGUE UPDATE"
490 PRINT:
    PRINT"enter:
     GOLFER OR TEAM NAME"
500 PRINT"    OR:
     <UF> IF UPDATE FINISHED"
510 PRINT"    OR:
     <MM> FOR MAIN MENU"
520 PRINT:
    LINEINPUT"  ";NM$
530 IF NM$="UF" THEN 950
540 IF NM$="MM" THEN 310
550 REM SEARCH FOR GOLFER OR TEAM NAME
560 FOR B=1 TO 56
570 IF NM$=NA$(B) THEN 600
580 IF NM$<>NA$(B) THEN NEXT B
590 IF NM$<>NA$(56) THEN CLS:
    PRINT NM$" not found":
    GOTO 490
600 IF TA(B,11)>0 THEN CLS:
    PRINT NM$" already entered":
     GOTO 490
610 PRINT:
    INPUT"POINTS WON.....";
    P
620 REM IF POINT INPUT WAS FOR A TEAM T
    HEN SKIP INPUT FOR GOLFER
630 IFB=1ORB=8ORB=15ORB=22ORB=29ORB=360
    RB=43ORB=50 THEN 690
640 INPUT"WHICH SIDE <F OR B>..";S$
650 IF S$="F" OR S$="B" THEN  660 ELSE
    640
660 INPUT"GROSS TOTAL.....";TA(B,9
    )
670 INPUT"NET SCORE.....";TA(B,1
    0)
680 INPUT"1ST HOLE SCORE.....";TA(B,1
    1)
690 PRINT:
    INPUT"ARE FIGURES OK <Y OR N>
    ";CK$
700 IF CK$="N" THEN TA(B,11)=0:
    GOTO 470
710 REM TOTAL THE POINTS FOR EACH GOLFE
    R OR TEAM AND TOTAL ALL INPUT
     POINTS
720 TA(B,8)=TA(B,8)+P:
    GT=P+GT
730 REM IF INPUT WAS A TEAM NAME THEN G
    O BACK TO UPDATE SECTION
740 IF CK$="Y" AND B=1ORB=8ORB=15ORB=22
    ORB=29ORB=36ORB=43ORB=50 THEN
     470
750 REM MOVE HANDICAP STORAGE POSITIONS
     IN THE ARRAY AND DELETE THE
    OLDEST
760 TA(B,7)=TA(B,6):
    TA(B,6)=TA(B,5):
    TA(B,5)=TA(B,4):
    TA(B,4)=TA(B,3)
770 REM CALCULATE THE HANDICAP FOR CURR
    ENT WEEK AND STORE IN ARRAY
780 IF S$="F" THEN TA(B,3)=(TA(B,9)-37)
    *.96
790 IF S$="B" THEN TA(B,3)=(TA(B,9)-35)
    *.96
800 REM IF THE HANDICAP EQUALS ZERO, AS
    SIGN IT A VALUE OF .0001
810 IF TA(B,3)=0 THEN TA(B,3)=.0001
820 REM FORMULAS TO CALCULATE THE HANDI
    CAPS DEPENDING ON NUMBER OF W
    EEKS PLAYED
830 IF TA(B,5)=0 THEN TA(B,12)=TA(B,2):
    GOTO 470
840 IF TA(B,6)=0 THEN TA(B,12)=(TA(B,3)
    +TA(B,4)+TA(B,5))/3:
    GOTO 470
850 IF TA(B,7)=0 THEN TA(B,12)=(TA(B,3)
    +TA(B,4)+TA(B,5)+TA(B,6))/4:
    GOTO 470
860 REM CALCULATION OF HANDICAP IF 5 HA
    NDICAPS ARE STORED IN THE ARR
    AY
870 N1=-10:
    N2=99
880 FOR E=3 TO 7
890 IF TA(B,E)>N1 THEN N1=TA(B,E)
900 IF TA(B,E)<N2 THEN N2=TA(B,E)
910 NEXT E
920 TA(B,12)=(TA(B,3)+TA(B,4)+TA(B,5)+T
    A(B,6)+TA(B,7)-N1-N2)/3
930 GOTO 470
940 REM SHOW EACH MAN IN THE FLIGHT PLU
    S NET AND 1ST HOLE SCORE
950 T=0
960 CLS:
    PRINT"DETERMINATION OF BALL W
    INNER"
970 PRINT:
    PRINT"NUM  GOLFER        NET
    1ST HOLE":
    PRINT
980 T=T+1
990 FOR B=1 TO 56
1000 IF TA(B,1)=T AND TA(B,11)<>0 THEN 1
    010 ELSE 1020
1010 PRINTUSINGV2$;B,P1$,NA$(B),TA(B,10)
    ,TA(B,11)
1020 NEXT B
1030 PRINT:
    INPUT"WHICH NUMBER IS BALL WI
    NNER";W
1040 BW$(T)=NA$(W):
    BS(T)=TA(W,10)
1050 IF T<6 THEN 960
1060 CLS:
    PRINT"BALL WINNERS      NET":
    PRINT
1070 FOR T=1 TO 6
1080 PRINTUSINGV3$;BW$(T),BS(T)
1090 NEXT T
1100 PRINT:
    INPUT"PRESS <ENTER> TO CONTIN
    UE";ZX$
1110 REM POSITION CHANGE SECTION
1120 CLS
1130 INPUT"IS THIS A POSITION CHANGE WEE
    K  <Y OR N>";PC$
1140 IF PC$="N" THEN 1420
1150 IF PC$="Y" THEN CLS:
    PRINTVC$:
    PRINT:
    GOTO 1180
1160 GOTO 1120
1170 REM PRINT ALL TEAM MEMBERS, POSITIO
    NS AND HANDICAPS
1180 FOR B=2 TO 56
1190 IFB=8ORB=15ORB=22ORB=29ORB=36ORB=43
    ORB=50 THEN 1230
1200 PRINTUSINGVB$;B,P1$,NA$(B),TA(B,1),
    TA(B,12)
1210 IF B=56 THEN 1230
1220 NEXT B
1230 PRINT:
    PRINT"ARE THESE POSITIONS / H
    ANDICAPS"
1240 INPUT"CORRECT <Y OR N>";CH$
1250 IF CH$="N" THEN 1300
1260 IF CH$="Y" AND B=56 THEN 1420
1270 IF CH$="Y" THEN CLS:
    PRINTVC$:
    PRINT:
    GOTO 1220
1280 GOTO 1230
1290 REM CHANGE THE POSITION ORDERS IN R
    ELATION TO THE HANDICAPS
```

```
1300 INPUT"HOW MANY TO CHANGE";NC
1310 FOR H=1 TO NC
1320 INPUT"NUMBER <2 TO 56> TO CHANGE";N
     P
1330 INPUT"CHANGE TO WHAT POSITION";NN
1340 TA(NP,1)=NN
1350 NEXT H
1360 CLS:
     PRINTVC$:
     PRINT
1370 IF B<=50 THEN FOR G=(B-6) TO (B-1)
1380 IF B>50 THEN FOR G=51 TO 56
1390 PRINTUSINGVB$;G,P1$,NA$(G),TA(G,1),
     TA(G,12)
1400 NEXT G
1410 GOTO 1230
1420 CLS
1430 REM COMPUTER SORTS TEAMS AND POINTS
1440 PRINT"TEAMS AND POINTS IN ORDER"
1450 REM ADD UP EACH TEAM'S POINTS
1460 AA=0
1470 FOR K=1 TO 50 STEP 7
1480 AA=AA+1
1490 LB(AA)=TA(K,8)+TA(K+1,8)+TA(K+2,8)+
     TA(K+3,8)+TA(K+4,8)+TA(K+5,8)
     +TA(K+6,8)
1500 IF WN<=8 THEN TA(K,7)=LB(AA)
1510 IF WN<=8 THEN TA(K,3)=TA(K,7)
1520 IF WN>8 THEN TA(K,6)=LB(AA)
1530 IF WN>8 THEN TA(K,3)=TA(K,6)-TA(K,7
     )
1540 L(AA)=TA(K,3)
1550 NEXT K
1560 REM PUT THE POINTS IN DECREASING NU
     MERICAL ORDER
1570 T=0
1580 Z=0
1590 Z=Z+1
1600 IF Z>8 THEN GOTO 1700
1610 IF L(Z)=-1 THEN 1590
1620 FOR J=1 TO 8
1630 IF L(J)>L(Z) THEN Z=J
1640 NEXT J
1650 T=T+1
1660 LL(T)=L(Z)
1670 L(Z)=-1
1680 GOTO 1580
1690 REM MATCH UP THE TEAM POINTS WITH T
     HE TEAM NAMES
1700 F=0
1710 F=F+1
1720 IF F>8 THEN 1810
1730 FOR M=1 TO 50 STEP 7
1740 IF TA(M,4)=1 THEN 1780
1750 REM ONCE NAMES AND POINTS HAVE BEEN
      MATCHED, SET A FLAG IN THE A
     RRAY
1760 IF LL(F)=TA(M,3) THEN H$(F)=NA$(M):
     TA(M,4)=1
1770 IF LL(F)=TA(M,3) THEN GOTO 1710
1780 NEXT M
1790 GOTO 1710
1800 REM PRINT TEAMS AND POINTS IN ORDER
1810 PRINT
1820 FOR U=1 TO 8
1830 PRINTUSINGV4$;H$(U),LL(U)
1840 NEXT U
1850 PRINT"----------------------"
1860 PRINT:
     PRINT"TOTAL POINT INPUT WAS"G
     T
1870 PRINT:
     INPUT"PRESS <ENTER> TO CONTIN
     UE";U$
1880 REM PRINT MENU
1890 CLS:
     PRINT"PRINT MENU"
1900 PRINT:
     PRINT"1. SAVE ON TAPE"
1910 PRINT"2. CHECK/CHANGE FIGURES"
1920 PRINT"3. PRINT OUT ALL FIGURES"
1930 PRINT"4. PRINT SCHEDULE"
1940 PRINT"5. END PROGRAM"
1950 PRINT"6. MAIN MENU"
1960 PRINT:
     INPUT"<ENTER> NUMBER";X
1970 IF X=6 THEN 310
1980 ON X GOSUB 2010,2200,2830,3080,4170
1990 GOTO 1890
2000 REM SAVE THE INFORMATION ON CASSETT
     E TAPE
2010 CLS
2020 PRINT"TAPE STORAGE SECTION"
2030 PRINT:
     INPUT"NAME OF FILE";NF$
2040 INPUT"READY RECORDER & PRESS ENTER"
     ;ZZ$
2050 CLS:
     PRINT"saving information on t
     ape"
2060 OPEN"O",#-1,NF$
2070 FOR A=1 TO 56
2080 PRINT#-1,NA$(A)
2090 NEXT A
2100 FOR B=1 TO 56
2110 FOR C=1 TO 12
2120 PRINT#-1,TA(B,C)
2130 NEXT C
2140 NEXT B
2150 CLOSE #-1
2160 CLS:
     PRINT"FILE DONE - UNDER NAME
     ";NF$
2170 PRINT:
     INPUT"BACK-UP FILE <Y OR N>";
     D$
2180 IF D$="Y" THEN 2010 ELSE RETURN
2190 REM SCREEN PRINT OF GOLFER OR TEAM
     TO CHECK OR CHANGE FIGURES
2200 CLS
2210 PRINT"CHECK/CHANGE FIGURES"
2220 PRINT:
     PRINT"1. INDIVIDUAL GOLFER/TE
     AM"
2230 PRINT"2. BACK TO MENU"
2240 PRINT:
     INPUT"<ENTER> NUMBER";CC
2250 ON CC GOTO 2270,2590
2260 GOTO 2200
2270 CLS
2280 INPUT"GOLFER/TEAM NAME";NM$
2290 FOR A=1 TO 56
2300 IF NM$=NA$(A) THEN 2330
2310 IF NM$<>NA$(A) THEN NEXT A
2320 IF NM$<>NA$(56) THEN PRINT NM$" not
      found":
     GOTO 2280
2330 CLS:
     PRINT NA$(A)
2340 PRINT:
     INPUT"IS NAME OK <Y OR N>";CG
     $
2350 IF CG$="Y" THEN 2370
2360 IF CG$="N" THEN INPUT"CHANGE NAME T
     O";NA$(A)
2370 IF A=1ORA=8ORA=15ORA=22ORA=29ORA=36
     ORA=43ORA=43ORA=50 THEN 2380
     ELSE 2450
2380 PRINT:
     PRINTUSINGV1$;" 1. TM#",TA(A,
     1)," 7. 1HP",TA(A,7)
2390 PRINTUSINGV1$;" 2. ---",TA(A,2)," 8
     . SUB",TA(A,8)
2400 PRINTUSINGV1$;" 3. 2HP",TA(A,3)," 9
     . ---",TA(A,9)
2410 PRINTUSINGV1$;" 4. FLG",TA(A,4),"10
     . ---",TA(A,10)
2420 PRINTUSINGV1$;" 5. ---",TA(A,5),"11
     . ---",TA(A,11)
2430 PRINTUSINGV1$;" 6. TPT",TA(A,6),"12
     . ---",TA(A,12)
2440 GOTO 2510
2450 PRINT:
     PRINTUSINGV1$;" 1. POS",TA(A,
     1)," 7. HC5",TA(A,7)
2460 PRINTUSINGV1$;" 2. SHC",TA(A,2)," 8
     . PTS",TA(A,8)
2470 PRINTUSINGV1$;" 3. HC1",TA(A,3)," 9
     . GRS",TA(A,9)
2480 PRINTUSINGV1$;" 4. HC2",TA(A,4),"10
     . NET",TA(A,10)
2490 PRINTUSINGV1$;" 5. HC3",TA(A,5),"11
     . HL1",TA(A,11)
2500 PRINTUSINGV1$;" 6. HC4",TA(A,6),"12
     . PHC",TA(A,12)
2510 PRINT:
     INPUT"IS INFORMATION OK <Y OR
      N>";CF$
2520 IF CF$="Y" THEN 2570
2530 IF CF$="N" THEN 2540 ELSE 2330
2540 INPUT"NUMBER TO CHANGE <1 TO 12>";E
     R
2550 INPUT"CHANGE THE INFORMATION TO";TA
     (A,ER)
2560 IF CF$="N" THEN 2330
2570 CLS:
     INPUT"ANOTHER GOLFER/TEAM <Y
     OR N>";AN$
2580 IF AN$="Y" THEN 2270 ELSE 2590
2590 RETURN
2600 REM LOAD FILE FROM CASSETTE TAPE
2610 CLS
2620 PRINT"TAPE LOADING SECTION"
2630 PRINT:
     INPUT"NAME OF FILE";NF$
2640 PRINT:
     PRINT"PRESS <PLAY> ON RECORDE
     R."
2650 INPUT"PRESS <ENTER> WHEN READY";EN$
2660 CLS:
     PRINT"scanning for "NF$
2670 OPEN "I",#-1,NF$
2680 CLS:
     PRINT"loading "NF$
2690 FOR A=1 TO 56
2700 INPUT #-1,NA$(A)
2710 NEXT A
2720 FOR B=1 TO 56
2730 FOR C=1 TO 12
2740 IF EOF(-1) THEN 2780
2750 INPUT#-1,TA(B,C)
2760 NEXT C
2770 NEXT B
2780 CLOSE #-1
2790 CLS:
     PRINT"FILE LOADED"
2800 FOR TT=1 TO 900:
     NEXT TT
2810 RETURN
2820 REM PRINT-OUT OF ALL INFORMATION IN
      BOTH LARGE ARRAYS
2830 CLS
2840 PRINT"PRINT OUT OF ALL INFORMATION"
2850 PRINT:
     PRINT"CHECK STATUS OF PRINTER
     "
2860 INPUT"PRESS <ENTER> WHEN READY";RY$
2870 PRINT#-2,"TEAM INFO    TM#  ---  2H
     P   FLG   ---   TPT   1HP   S
     PT  ---  ---  ---    ---"
2880 PRINT#-2,"  OR"
2890 PRINT#-2,"GOLFER       POS  SHP  HC
     1   HC2   HC3   HC4   HC5   P
     TS  GRS NET HL1    PHC"
2900 FOR B=1 TO 56
2910 PRINT#-2,USINGV5$;NA$(B),TA(B,1),TA
     (B,2),TA(B,3),TA(B,4),TA(B,5)
     ,TA(B,6),TA(B,7),TA(B,8),TA(B
     ,9),TA(B,10),TA(B,11),TA(B,12
     )
2920 NEXT B
2930 RETURN
2940 REM SET FIRST HOLE FLAGS AND TEAM P
     OINT ORDERING FLAGS TO ZERO
2950 CLS:
     PRINT"ABOUT TO SET FLAGS TO Z
     ERO."
2960 INPUT "<Y OR N>";Y$
2970 IF Y$="Y" THEN 2980 ELSE 3060
2980 CLS:
     PRINT "1ST HOLE AND FLAGS SET
      TO ZERO"
2990 FOR B=1 TO 56
3000 TA(B,11)=0
3010 NEXT B
3020 FOR B=1 TO 56 STEP 7
3030 TA(B,4)=0
3040 NEXT B
3050 FOR TT=1 TO 1000:
     NEXT TT
3060 RETURN
```

```
3070 REM PRINT-OUT OF WEEKLY PLAYING SCH
     EDULE
3080 CLS:
     PRINT"WHICH MATCH NUMBER IS T
     HIS"
3090 INPUT"PRINT-OUT FOR";MN
3100 PRINT:
     PRINT"SCHEDULE FOR FRONT OR B
     ACK?"
3110 INPUT"TYPE IN <FRONT or BACK>";SS$
3120 PRINT:
     PRINT"DO YOU WANT BALL WINNER
     S AND"
3130 PRINT"TEAM STANDINGS TO PRINT OUT?"
3140 INPUT"<Y OR N>";EP$
3150 CLS:
     PRINT"CHECK STATUS OF PRINTER
     "
3160 INPUT"PRESS <ENTER> WHEN READY";RY$
3170 PRINT#-2,"  LIVERPOOL BUSINESSMAN'S
      GOLF LEAGUE          MATCH NU
     MBER";MN;" ";SS$;" SIDE"
3180 FOR XX=1 TO 4:
     CLS
3190 PRINT"SCHEDULE SECTION NUMBER ";XX
3200 PRINT"ENTER IN THE NUMBERS OF THE T
     WO"
3210 PRINT"TEAMS THAT ARE TO PLAY EACH"
3220 PRINT"OTHER (IE. 11,22)."
3230 INPUT T1,T2
3240 REM TRAP FOR TEAM NUMBER INPUT
3250 FOR TR=11 TO 88 STEP 11
3260 IF TR<>T1 THEN NEXT TR ELSE 3280
3270 GOTO 3190
3280 FOR TR=11 TO 88 STEP 11
3290 IF TR<>T2 THEN NEXT TR ELSE 3320
3300 GOTO 3190
3310 REM LOCATE FIRST TEAM FOR PRINT-OUT
3320 FOR A=1 TO 56
3330 IF TA(A,1)=T1 THEN 3370
3340 NEXT A
3350 IF TA(A,1)<>T1 THEN 3190
3360 REM ARRANGE GOLFERS IN THE FIRST TE
     AM BY POSITION
3370 FOR Q=1 TO 6
3380 FOR R=1 TO 6
3390 IF TA(A+R,1)=Q THEN PP(Q)=A+R:
     GOTO 3410
3400 NEXT R
3410 NEXT Q
3420 REM LOCATE SECOND TEAM FOR PRINT-OU
     T
3430 FOR B=1 TO 56
3440 IF TA(B,1)=T2 THEN 3480
3450 NEXT B
3460 IF TA(B,1)<>T2 THEN 3190
3470 REM ARRANGE GOLFERS IN THE SECOND T
     EAM BY POSITION
3480 FOR Q=1 TO 6
3490 FOR R=1 TO 6
3500 IF TA(B+R,1)=Q THEN PQ(Q)=B+R:
     GOTO 3520
3510 NEXT R
3520 NEXT Q
3530 REM PRINT FIRST TEAM VS SECOND TEAM
      PLUS HANDICAPS AND TOTAL POI
     NTS
3540 PRINT#-2,CHR$(10)
3550 PRINT#-2,"TEAM / NAME          HCP
      POINTS....VS....TEAM / NAME
           HCP    POINTS"
3560 PRINT#-2,USING V6$;NA$(A),TA(A,8),N
     A$(B),TA(B,8)
3570 FOR Q=1 TO 6
3580 PRINT#-2,USING V7$;Q,P1$,NA$(PP(Q))
     ,TA(PP(Q),12),TA(PP(Q),8),Q,P
     1$,NA$(PQ(Q)),TA(PQ(Q),12),TA
     (PQ(Q),8)
3590 NEXT Q
3600 REM PRINT TEAM TOTAL POINTS FOR FIR
     ST OR SECOND HALF OF SEASON
3610 IF WN<=8 THEN PRINT#-2,USINGVA$;P2$
     ,TA(A,3),P2$,TA(B,3)
3620 IF WN>8 THEN PRINT#-2,USINGVA$;P3$,
     TA(A,3),P3$,TA(B,3)
3630 NEXT XX
3640 PRINT#-2,CHR$(10)
3650 REM PRINT BALL WINNERS AND TEAM STA
     NDINGS IF WANTED
3660 IF EP$="N" THEN 3750
3670 PRINT#-2,"BALL WINNERS          SCOR
     E                TEAM STANDIN
     GS        POINTS"
3680 FOR V=1 TO 6
3690 PRINT#-2,USINGV8$;V,P1$,BW$(V),BS(V
     ),V,P1$,H$(V),LL(V)
3700 NEXT V
3710 FOR V=7 TO 8
3720 PRINT#-2,USINGV9$;V,P1$,H$(V),LL(V)
3730 NEXT V
3740 REM PRINT ANY SPECIAL MESSAGES AT T
     HE BOTTOM OF THE SCHEDULE
3750 CLS:
     PRINT"ANY SPECIAL MESSAGES TO
      PRINT?"
3760 INPUT"ENTER <Y OR N>";ME$
3770 IF ME$="Y" THEN 3780 ELSE RETURN
3780 PRINT:
     PRINT"ENTER ANY MESSAGE UP TO
      80"
3790 PRINT"CHARACTERS IN LENGTH."
3800 PRINT:
     LINEINPUT SM$
3810 PRINT#-2,SM$
3820 GOTO 3750
3830 REM INITIAL INPUT OF GOLFERS AND TE
     AM NAMES
3840 FOR A=1 TO 56
3850 CLS
3860 PRINT"INITIAL INPUT OF DATA"
3870 IFA=1ORA=8ORA=15ORA=22ORA=29ORA=360
     RA=43ORA=50 THEN 3880 ELSE 39
     20
3880 PRINT:
     PRINT"POSITION IN ARRAY="A
3890 PRINT:
     INPUT"TEAM NAME.....";NA$(A)
3900 INPUT"TEAM NUMBER....";TA(A,1)
3910 NEXT A
3920 PRINT:
     PRINT"POSITION IN ARRAY ="A
3930 PRINT:
     INPUT"GOLFERS NAME...";NA$(A)
3940 INPUT"POSITION.......";TA(A,1)
3950 INPUT"HANDICAP.......";TA(A,2)
3960 TA(A,12)=TA(A,2)
3970 NEXT A
3980 RETURN
3990 REM PRINT USING ROUTINES
4000 P1$="."
4010 P2$="1st HALF TOTAL......"
4020 P3$="2nd HALF TOTAL......"
4030 V1$="%      %###.##  %      %###.##
     "
4040 V2$="#### %             %##    ##"
4050 V3$="%            %   ##"
4060 V4$="%            %###.#"
4070 V5$="%           % ##  ## ###.#  ##
     .#  ##.# ###.# ###.#  ###.#
     ##  ## ##  ##.#####"
4080 V6$="%            %            ###
     .#          %          %
               ###.#"
4090 V7$="#####          %  ##     ###
     .#        #####
     %  ##     ###.#"
4100 V8$="#####          %    ##
                #####
       %   ###.#"
4110 V9$="
                #####
       %   ###.#"
4120 VA$="  %                  %###.#
                 %
          %###.#"
4130 VB$="#####           %##  ##.##"
4140 VC$="NUM GOLFER       POS  HCP"
4150 RETURN
4160 REM END THE PROGRAM
4170 CLS:
     PRINT"ABOUT TO END THE PROGRA
     M"
4180 INPUT"<Y OR N>";ED$
4190 IF ED$="Y" THEN END ELSE RETURN
```

Tandy topics

For all readers

Ed Juge, Director of Merchandising, Business Computer Products
1500 One Tandy Center, Fort Worth, TX 76102

The January issue of *Basic Computing* contains an interview with Chuck Tesler. Chuck indicated we might take exception to some of his comments, and I wouldn't want to disappoint him! I can't argue SuperScripsit with him, it *was* very late! We expected to have it by the time the first advertising appeared, but our choice was to slip it or bring it out with known bugs. I don't think you'll see much more "early advertising" from us.

The gentleman simply isn't well informed on us and our attitudes. His comment that in our viewpoint there are two types of people, us and deadly enemies, just doesn't wash with the more than 1,200 software companies we're working with in our Outside Software Support Program. No, we haven't taken on products that are competitive to our own. That wasn't the purpose of the program. It was set up to fill gaps in our software line. Apple didn't ask us to let them sell the Tandy 2000 either. It hardly means they're "predatory" just because they choose not to sell our products. Come on, Chuck, join the real world!

Yes, we said we might accept some (not just "any") products for private-labeled, in-store sales. So many have been submitted, I'm afraid we're way too far behind in replying. The good news is, as I've indicated before, our software strategy is undergoing quite an overhaul, for several reasons:

1) We can't afford to inventory everything customers want at the store level.

2) There's a practical limit to the number of programs our store personnel can learn to demonstrate, much less support.

3) Ditto our customer service and programming staffs.

4) We need to respond to the ever-growing demand for, and supply of, programs in the marketplace.

5) There's also a limit to the number of packages we can afford to inventory at the warehouse level.

Our solution will go something like this: Under the Radio Shack label, and available in our stores, expect to find the Word Processing, Spreadsheet, Database Management, Accounting, Language, Game, Home Education, and some other "Productivity" programs. Available on customer-order only (not store stocked), will be Radio Shack branded communications, development, school, and a few vertical market packages among others. For some time now, we've had categories of non-store stocked programs, which aren't fast enough movers to justify shelf space.

What's "Express Order Software?"

Effective before you read this column, we will have initiated a program tentatively called "Express Order Software." It will consist of software stocked in the warehouse only, and available for overnight shipment to any store when a customer orders it. Express Order Software products will *not* be Radio Shack branded. Instead, they would carry the Micro-Pro, Peachtree, Digital Research, or whatever, label. (I am not promising those lines, just an example). They will not be supported in any way by Radio Shack. If you have a question, there will be a customer service phone number in each package for you to call. It will be that of the author/publisher. Our store and customer service people won't be familiar with the operation of these items. We will be acting as a dealer or sales agent only. For those of you who are familiar with it, Express Order Software will be similar to IBM's "Non-Logo" software program.

Wow, tens of thousands of programs! Right? Wrong! We will probably stock almost anything which has proven itself to be a market leader (top 50 seller list, etc.), if the publisher wants to participate. We'll carry *good* vertical market software. Each item will be stocked in *small* quantities until it proves itself, then whatever is needed to keep the pipeline full. And yes, Chuck, Express Order Software finally gives us a viable way to offer competitive products!

There will be a selection process, and offerings will still be rejected. We will publish an Express Order Software program catalog, probably quarterly. We're excited about it, as are the vendors we've discussed it with. As I write this column, we're ironing out the final details of the contract. Submissions should be through Phil Kitchen, Manager of our Outside Software Support Group, 1300 One Tandy Center, Ft. Worth, TX 76102.

Support of Older TRS-80's

This subject has been covered before, but it's been cropping up more often lately, with the introduction of the Tandy 2000. It's probably time to cover it again.

A gentleman recently called several people in Tandy Center, complaining of lack of support for his Model I. He said that we said that we'd support it forever, and we aren't. Well, "forever" is probably an overstatement. Am I prejudiced? Probably, 'cause in my den still sits a 16K, Level II, non-disk Model I. I've refused to sell it, yet I don't use it any more. Since it caused a major change in my career path, I'm sentimentally attached. If it wasn't so all-fired big, I'd probably have it bronzed!

Anyway, we did say we would

continue support for discontinued models as long as there was sufficient demand. Model I owners will attest that we split out their disks from some of the I/III compatible packages, so that in addition to the package, we had to separately order the Model I disks for them. That, among other things, gave us a fair barometer of demand for Model I packages. All of our other indicators tell us that demand has waned to a very low level, even though there are hundreds of thousands of Model I's still in use.

Second, some packages simply won't fit in the 32K available memory, or on the standard Model I disks. For whatever reasons, we don't believe we can 'downsize' the capabilities of an item, just to make it backward-compatible. It isn't fair to the owners of newer models, and would make us highly uncompetitive in the marketplace. As technology lets us do grander things, we must take advantage of those opportunities.

A substantial amount of our software is written by third parties who receive royalties on sales. Many of them choose not to devote their resources to developing versions compatible with anybody's non-current computers. So, backward compatibility is most often a matter of the cost in terms of resources, time, and program capabilities.

As to hardware, that's another matter. Model I owners have asked us to build Expansion Interfaces. The FCC specifically limited the number of them we could build during the year following the emission regulations. Sorry, we can build no more. Other hardware items we tried met with little demand. Am I saying we're through supporting Model I (or III or II, etc.)? By no means. I am explaining the factors which determine the level of support we will offer.

The Elusive CP/M!

Maybe I'm safe in saying CP/M will be available by the time you read this. It has been released for production, for both the Model 4 and the Models II/12. Model 12 owners can address the full 80K of RAM. Model II will be to 64K. Model 16 conversions will still only address the Z-80 RAM as outlined above.

Things are going well. Model 4P's and Tandy 2000's are in short supply due to parts availability, but the situation should improve soon. Multi-user Scripsit and Profile are in the final stages of testing, and could make it to the warehouse about the time you get this magazine. Most of the Tandy 2000 initial tier of software is available as of this writing. Accounting was delayed by a hiccup in the printer driver, but should be corrected next week. Word Processing (MultiMate) is in test, looking good, waiting the remaining parts of documentation to come in from the vendor. It is targeted for release February 15th. Some very exciting software is in the queue for the 2000! Stay tuned, see you next month. (Boy, it's nice to be able to transmit my column to *Basic Computing* via EMAIL! Federal Express, eat your heart out!)

BASIC bits

Model 4 tips and more on running in Model III mode

Model III/4

Thomas L. Quindry

I'm very new to computing. In the article, "Exploring VisiCalc" (November, 1983 issue, page 59), Timothy Bowman refers to a Model 4 running in the Model III mode. How can this be done? Is it documented in either the owner's manual or the technical manual, or is it the product of some superior programmer's software program which I do not have or know about? Also, how can I POKE graphic characters to the screen in BASIC? The manual "Getting Started With TRS-80 BASIC" for Models I and III gives the information for those computers, but it won't work on the Model IV.

R.S., LaCygne, KS

The Model 4 has a dual operating system to allow compatibility with older models. Surprisingly, the Model 4 manual does not convey this information properly. Someone new to the Radio Shack computer line would have to stumble on this by accident as a friend of mine did. One mode of operation is as a Model III (including cassette operation). On power-up, or when rebooting (pushing the reset button), you can use either a Model III or Model 4 Disk Operating System (DOS). TRSDOS 6.0 comes with the Model 4 as the Model 4 DOS. Any of the Model III DOSs on the market today will automatically cause your Model 4 to run in the Model III mode.

To use the Model III mode, buy a Model III DOS and boot up the disk. The least expensive is Radio Shack's TRSDOS 1.3 for $14.95 and the Model III Disk Systems Owner's Manual for $6.95. Since not much in the way of software yet exists for the Model 4, it wouldn't be a bad idea to have the Model III DOS anyway. You can also hold down the break key while hitting the reset button and get into Model III Level II BASIC. This does not support disk but you can load programs by cassette.

You cannot POKE to the video screen in the Model 4 mode. The Model 4 video can only be written to by what is called Service Calls. The BASIC PRINT command does this. You should consult your technical manual to determine how to do this in machine language. The reason POKE does not work is because the video memory is bank-switched. There is no place for you to POKE to since video memory is never accessible when you want to POKE.

Now that we have a Model I, Model III and Model 4 that are upward-compatible, those of us who write programs to run on any of the machines need a way to know which machine is in use and what the clock speed is.

I understand POKE 16912, PEEK (16912)OR64 will make the Model 4 run at 4Mhz in the Model III mode, but it seems to make TIME$ run at twice speed. Is there a way to make the Model 4 run at 4Mhz in the Model III mode and still have correct real-time clock operation?

Finally, is there a way to access the internal sound from the Model III mode? It would be nice if there were an easy way to modify the many Model III programs that output sound through the cassette port to run on the Model 4 without requiring an external audio amplifier.

J.S., Madison, WI

I'll answer the questions in reverse order. To get your answers, I called on users of our computer club bulletin board, the TCUG resource network. Amir Naini of Huntington, MD responded with an answer to the sound output question. Amir says that you just change the output port used in your cassette port sound generation program from the cassette port 255 to port 144. It is his opinion that the inexpensive external speaker that Radio Shack sells for $11.95 (cat. no. 277-1008) sounds better than the speaker installed in the Model 4 so you are better off not changing the port address.

Your solution to Model III mode clock speed-up is only partly correct. John Harrell of Washington, DC gives the following:

```
10 X=PEEK(16912)
20 X=X OR 64
30 POKE 16912,X
40 OUT 236,X
```

The output to the port is necessary to correct disk I/O operations. (Port

236 will eventually get changed to the correct value by normal computer operations by things such as CLS but you might as well set it straight at the outset.) John also gives the following to slow back to the Model III clock speed:

```
10 X=PEEK(16912)
20 X=X AND 191
30 POKE 16912,X
40 OUT 236,X
```

Though it shouldn't be, I must caution you that it has been reported that there still may be some disk I/O problems *(see last issue's "Notes, etc." column —Ed.)*. So, it may be a good idea to use the slow-down procedure given just before accessing the disk. You can speed it up again just after accessing.

The essence of the above two routines is that bit 6 of the address and port given is set or reset depending on clock speed desired.

Now, the hard part. My thanks to Jerry Weisskohl of Falls Church, VA for letting me try out his Model 4 to test the following. Running the Model III mode with the clock speed-up does, in fact, accelerate the values placed in the binary time-date addresses at 16919 through 16924 decimal. This is because outputting the speed-up bit to the port causes twice as many interrupts to be made. This was done by design, not by mistake, for Model 4 operation. The clock-pulse counter has to be made to double the count for each second. The code starting at 3529H, among other things, decrements the clock-pulse counter to zero, increments the clock buffer, and resets the pulse counter with a command to LD (HL),01EH starting at address 355CH.

In the Model III mode of the Model 4 with the clock speeded up, this value of 01EH must be increased to twice the value, or 03CH, to cause the clock to count correctly. Since this code is in ROM, I have written a patch code routine. The routine I've written changes the address given at 4047H to send control to my routine, repeats all code from the address 3529H to 355CH, and gives a different value than that given in 355DH to set the maximum number of clock pulses needed to increment the clock by one second. It then jumps back to the ROM routine starting at 355EH.

One word of caution. Your natural impulse will be to name this routine, CLOCK/CMD. Don't do it. If you try to access the program by entering the word CLOCK, you will only turn on the clock and date display in the upper right-hand corner of your screen. Better yet, name it CLOCKSPD/CMD.

My June, 1983 column gives a table which will tell you which computer you are on and whether it is cassette- or disk-based. It refers to the Model I and Model III. (Note the correction given in September, 1983.) These values hold for the Model III mode of the Model 4 as best I can determine. Since the Model 4 BASIC is different, as well as the screen size, it would probably be best to write a different program for the

Model 4 mode. In the Model III mode of the Model 4, you can determine the clock speed being used by X= PEEK(16912)AND64. If the value returned is 64, the clock has been speeded up. Otherwise, the value returned will be 0 and it will be the regular Model III mode clock speed.

Roy Soltoff, the primary designer of the TRSDOS 6.0 operating system, has told me that C version ROM of the Model III and Model 4 differ; mainly because of initializing that has to be done because of Model 4-specific operations. I have made a comparison of these ROMs and have come up with what I think is a failsafe way to determine whether you are on a Model III or Model 4 computer (Model III mode). The Model III printer driver translation table was in error, either by design, or whatever. If you were to LPRINT CHR$(96) on the Model III, you would get the @ symbol. You should have gotten the backslash symbol instead. Using OUT251,96 will give you the correct character. In programming, PEEK(&H3185) should give you a 96 with the Model 4 in Model III mode, the corrected value, and a 64 with the Model III. Add that to my table of values given in the June, 1983 issue of *Basic Computing*.

Remember to send your requests for future column topics, questions and tips to me, care of *Basic Computing*, 3838 South Warner Street, Tacoma, WA 98409. Send a self-addressed, stamped envelope and I'll try to give you a personal reply. Problems of general interest may be included in future "BASIC bits."

Listing 1 — CLOCKSPD/CMD

```
00100 ;MODEL IV @ MODEL III MODE CLOCK SPEED CHANGE
00110         ORG     0FFA0H
00120 BEGIN   LD      HL,CLOCK
00130         LD      (4047H),HL      ;CHANGE VECTOR ADDR.
00140         LD      A,(4210H)       ;SPEED UP CLOCK
```

```
00150           OR      40H
00160           LD      (4210H),A
00170           OUT     (0ECH),A
00180           LD      HL,CLOCK-1      ;SAVE HI-MEM
00190           LD      (4411H),HL
00200           JP      402DH           ;JUMP TO DOS
00210 ;SUBSTITUTED CLOCK ROUTINE FOLLOWS
00220 CLOCK     LD      DE,3591H
00230           PUSH    DE
00240           IN      A,(0ECH)
00250           LD      A,(4022H)
00260           OR      A
00270           JR      Z,CLKBIT
00280           LD      A,(401CH)
00290           OR      A
00300           JR      NZ,CLKBIT
00310           LD      HL,401AH
00320           DEC     (HL)
00330           JR      NZ,CLKBIT
00340           LD      (HL),7H
00350           INC     HL
00360           LD      A,(HL)
00370           AND     01H
00380           XOR     01H
00390           LD      (HL),A
00400           LD      HL,(4020H)
00410           JR      Z,SPACE
00420           LD      A,(4023H)
00430           JR      CHAR
00440 SPACE     LD      A,20H
00450 CHAR      LD      (HL),A
00460 CLKBIT    LD      HL,4216H
00470           DEC     (HL)
00480           RET     NZ
00490           LD      (HL),03CH       ;CLOCK PULSE COUNTER
00500           JP      355EH
00510           END     BEGIN
```

Software protection

Notes from a lecture. What does © mean?

For all readers

Kennerley C. Ashley, D.D.S.

The following are notes on a lecture by Naveed Alam, Attorney at Law, at the March 20, 1983 meeting of the Orange County TRS-80 Users Group. Mr. Alam is a member of the law firm of Weissenberger and Peterson in Newport Beach, California, and specializes in patents, trademarks and copyrights. The notes were taken by Ken Ashley, Orange County TRS-80 Users Group Secretary.

I. A new Copyright Law was passed in 1978 (17 United States Code §101). Prior to that time, it was unclear whether copyright protection was possible for computer programs because it was felt that copyright covered only "writings" which communicate between persons, and not person to machine. In 1980, the Computer Software Act was passed. It makes clear that copyright protection extends to software. This is a new field in law and there is a lot of grey area. There are few court decisions to test the law. We are using precedents in law that were set for paintings, books and movies. Mr. Alam has his opinions as to what the Act means, but even he and his boss don't agree.

Copyright protection, from its inception, has been only for the expression of an idea and not the idea itself. The expression has to be original and in a tangible form. A lecture being presented cannot be copyrighted until it is reduced to tangible form (i.e., taped, recorded, or written down). A computer disk is a tangible (copyrightable) form, but a program running in the computer is not. Excluded from copyright protection are: ideas, phrases, titles, blank forms, measuring charts, discoveries, etc.

A computer disk is a tangible (copyrightable) form, but a program running in the computer is not!

II. To get copyright protection, you must give notice that you intend your work to be protected. Until it is "published" it need not bear any copyright notice. "Published" has a special legal meaning. Generally, unrestricted distribution implies publication. One of the few court decisions in the copyright area held that the distribution of 200 copies to a limited membership was not publication because the distribution was to a controlled group. If you sell or even give away your program, that is publication. If you want it protected, it must bear the copyright notice: ©, the word "Copyright," or the abbreviation "CPRT." along with the name of the copyright owner and the year of first publication.

The author may not be the owner. For example, if an employee develops a program in the course of his employment or even at home and it relates to his employer's business, the employer owns the program unless the employment contract states otherwise. Ownership of a program can be sold or transferred. If it is a joint work, both names should appear on the copyright notice. The notice should be placed at the beginning of the program listing, on the main screen, on the disk label and in the manual. Since the law is so new, you want to cover all possibilities. If your program produces substantial sound, you should cover that as a phonograph recording with the symbol "P" in a circle. If there is unique visual display or printout, they can be protected also, apart from the computer program itself.

Omission of copyright notice can be cured. For example, you can write a letter to those who purchased your program without the notice and inform them that the omission was an error, and your intention to claim copyright. You should document your efforts to meet any challenge later.

III. To register your copyright, you should send a copy of the program listing to the Copyright Office along with the registration fee of ten dollars. Registration is

optional, but provides certain benefits and advantages. Separate and apart from registration, the Copyright laws contain a deposit requirement. Within three months after your program is "published," you must deposit a copy of it with the Copyright Office. The purpose is to allow the Copyright Office access to select work that should be preserved for archival purposes. Failure to meet the deposit requirement for archival purposes does not affect your copyright. If you don't send them a copy and the Copyright Office decides they want your work, they can buy it on the open market and fine you. The deposit for archival purposes can be combined with the deposit required for registration for copyright. Registration is optional, except that you must register your copyright before filing any law suit for infringement. Your registration certificate is presumed to be valid in a suit, making the burden of proof fall upon the challenger.

IV. The Copyright Office has certain forms that must be used. Form TX is for literary works, that includes computer programs; form PA for audio-visual and movies; form SR for sound recordings or melodies; and VA for pictures, graphics or works of art. You use a different one for each aspect of the program you wish to copyright. If your program produces certain original melodies and prints out a nice original graphic, you can use form TX for the computer program itself, form SR for the melody, separate and apart from the program, and form VA for the graphics separate and apart from the program. Each form is sent with a ten dollar fee. You can do this yourself without the help of an attorney. You should be aware that the Copyright Office may refuse registration for the output of your program on the ground that the "author" of the output is the computer, not the individual. You should contest this. Mr. Alam disagrees with the Copyright Office. There are ways of getting around the Copyright Office's position.

You can register a number of your programs as a "collection." The collected volume should be work by one author or owner, and have one title. Contributions to periodicals over a twelve-month period can be registered as a collection.

The law of copyrighting computer chips is unclear. You can try to register them as a sculptural work or the enlarged photograph of the artwork of their design as a graphics work.

V. The author or owner has the following rights to his program for his lifetime plus fifty years: to reproduce, to distribute publicly, to input into a computer, to adapt, to translate, to produce derivative works, to perform publicly, to display, to reproduce in another medium, and to its use on time share networks.

The author's rights are limited by the following rights of the lawful possessor of the program: The owner has the right to adapt the program to fit his needs, to input it into his computer and run it, to sell his rights and transfer it to someone else, and to make secondary copies for archival purposes. It is considered fair use for a reviewer to reproduce small excerpts in his review. Special rules of use apply to libraries, schools, churches, and for the use of handicapped persons.

VI. Loaning or exchanging copies with friends is not an infringement because it is not "public distribution." If you have five hundred "friends," that may push things too far.

To prove infringement, you must show access and substantial similarity. If you can't prove access, there is no infringement even if there is total similarity. Because of legal costs, only large volume infringements get to court. You can recover statutory damages even where you have suffered no loss. Statutory damages range between $250 to $10,000, depending on whether the infringement is innocent or willful. You can sue for lost profits, the infringer's profits, your lawyer's fees, and court costs in your suit.

In criminal infringement cases you are in the "big league" — say, sales of fifty or sixty thousand copies. Here, the United States government would sue you. For criminal copyright infringement, there may be a fine of $200,000 and forfeiture of all equipment used.

VII. Along with copyrights, you should consider the protection provided by design patents and trademarks. Usually, when there is copyright infringement, you can additionally sue for unfair competition, interference with business, mental distress and anything else the attorney can think of.

VII. Question and answer period: If you hire a programmer, the work he produces is yours. Joint buyers of a program share equally in possession rights. If you have twenty computers in your office, you must buy twenty copies of a program to be legal. You may sell a program and the rights to it, but you must destroy all backup copies. You should not even give them to the buyer unless the copyright owner consents. Owners of programs have a right to make backup copies for "backup" purposes, not as a substitute for purchasing additional copies. When you make improvements in your copyrighted program, you should file a new copy with the Copyright Office.

A plea for organized programming

How to write code for a compiler

For all readers

Loyd Bulmur, Primero Consultants, Inc.

Notwithstanding the assertion by Richard Kaplan in the June, 1983 issue of *Computronics Magazine* that "It is not possible to write a structured program in BASIC," it is possible to bring a degree of organization to a BASIC program which borders on the "structured" concept. And it will reduce the number of errors encountered when using a compiler, as outlined in Spencer Koenig's column in the same issue.

This is especially true when a program is being written for publication, when the author should take into account the end-user, the person who will be keying in the published program, and the utilities which may be available.

Comment (REM) statements are very valuable in explaining the program's workings. But not all end-users may wish to include them in their keyed-in versions for various reasons (lack of RAM, personal preference, etc.). Please, then, do not direct GOTO or GOSUB statements to a line number which consists only of a REM. If the end-user removes the REM lines, the familiar error message "Undefined Line Number" will crop up at run-time and cause problems.

Line numbers should be incremented by 10. There is the chance that some end-user may come up with an enhancement or modification that could be inserted between existing line numbers. While many end-users have line renumbering utilities and can do this, why impose this onus on them? And why penalize those who don't?

As Mr. Koenig pointed out in his column, there is a growing use of compilers. Nearly all of those currently on the market impose some degree of organization. For instance, a GOSUB from a larger line number to a smaller one is unacceptable. While this is fine for the BASIC interpreter which follows instructions from one statement to the next, including conditional and unconditional branches, when the compiler tackles a BASIC program, it works from the smallest line number to the largest. If it runs into a GOTO 10000 on line 10, unlike the interpreter, it does not immediately do so. Instead, it tucks into memory that there is a GOTO to that line, and proceeds to the next line after 10. If, on that next line, it finds a RETURN (having been generated by a GOSUB from line 10020 which, in turn, has been accessed through the GOTO 10000 previously mentioned) it reports an error RETURN WITHOUT GOSUB.

A few years ago, there was a (mistaken) school of thought which said to put all your repeated subroutines at the beginning of your program. The computer needs less time (the argument went) to interpret a line number of 200 than of 20000. In fact, this is not true in compiled programs. When a BASIC program is saved, the computer interprets it into compressed binary and all line numbers are stored in hexadecimal. Each line number takes two bytes in hex no matter what the decimal equivalent is. Line number 10 is stored as 0A 00 hex and line 20000 is stored as 4E 20. Indeed, there is no saving on bytes, therefore no speedier execution.

Similarly, the END statement should always be the last numbered line in a program. If it's tucked away in the middle of a program, when the compiler reaches that line and finds END, it says "that's all folks" and stops compiling, leaving out all the good things which may follow that instruction.

Some compilers require that the source program which is to be compiled, be in ASCII format. Therefore, do not exceed 255 bytes per line. When the interpreter saves the program in compressed binary, it compacts such statements as GOSUB into a one-byte "token." But in ASCII, that statement used five bytes. It is possible to have a program line containing 255 bytes in compressed binary but it will probably exceed 255 bytes in ASCII.

Be careful with FOR . . . NEXT loops. Most compilers will recognize only one NEXT for each FOR, although an interpreter will permit multiple NEXT's for one FOR.

Say, for example, you are running a search on a random-access disk file for all occurrences of the word COMPUTER. Your code may look something like this:

```
100 FOR K = 1 TO LOF (1)
110 GET 1,K
120 IF A$<>"COMPUTER"THEN
NEXT
130  PRINT "AHA, FOUND YOU!
RECORD #"; K
140 PRINT "NOW LOOKING FOR
ANOTHER": NEXT
```

That code will work fine with an interpreter, but will generate an error when compiled since there are two NEXTs for one FOR.

The same prohibition holds true for GOSUB . . . RETURN, with one important difference. There can be unlimited GOSUB statements calling the same subroutine but there can be only one RETURN.

Another aid to organized programming is to keep a list of variables and what they represent with each program. This can take the form of a printout if you have a printer, stored on disk or tape, or a hand-written list. When you go to enhance a program you had written a year or so previously, many programmers (particularly active ones) have difficulty recalling what various variables stand for.

Another aid to organized programming is to use the same blocks of line numbers for repeated things. Such a prestigious firm as Digital Equipment Corp. suggests in its user guide that a structure be set forth and adhered to such as: program lines 1-999 be reserved for REM comments on the history of the program (who wanted it, why was it needed, what does it do). This is to help different programmers who may work on the program over the course of a number of years. While that is fine for computers with a lot of disk space and high amounts of core (RAM), it isn't necessary for us little guys to go that far. Perhaps keeping a written off-line record would not be too bad, either.

To get back to the concept of allocating blocks of line numbers, you could start the main line of execution at line 100, using the preceding line numbers for such things as authorship, date of revised version and so forth, then using a block of lines at 20000 for OPEN disk files, followed by FIELD statements. LPRINT subroutines could similarly be grouped together. This also helps in debugging. If you have five or six LPRINT subroutines and one does not work properly, you immediately know where to go in the program to effect the changes, and you have the advantage of having the other subroutines close by so a line-by-line comparison on screen is simpler.

As you write more and more programs (and periodically enhance them as your skills at programming increase), having similar sets of instructions at similar line numbers becomes a help in implementing enhancements. One does not have to search the listing to discover where the printer subroutines are located in such a program. Automatically, the programmer can go directly to, say, line 15000 and know that is the start of the printer subroutines. By using the same variable in different programs (e.g., INPUT "IS A PRINTER COPY REQUIRED Y/N "; Q$), it becomes easier to debug and enhance programs.

This is not to suggest that every programmer should use the same variable names. This is to recommend that each programmer should consider organizing the work in such a way that it becomes easier for the programmer (and the end-user) to introduce variations and enhancements.

End of plea. The defense rests.

Reviews

LDOS Wordstar with Mailmerge
Model III/4III
Logical Systems, Inc.
8970 N. 55th Street
Milwaukee, WI 53223
(414) 355-5454
$395.00

LDOS Wordstar allows the Model III user an opportunity to utilize the famous (or infamous) Wordstar word processing system. With this software, Wordstar may be run *without* requiring a CP/M modification to the computer.

Wordstar was originally written to be used with the CP/M operating system. Logical Systems, Inc. has altered the program to make it compatible with their LDOS operating system. This version of Wordstar has *all* of the standard features except two: automatic generation of a backup file, and the ability to run a command file from within Wordstar. Neither of these omissions is particularly significant.

Wordstar is the *de facto* standard by which current word processors are judged. It is not, by far, an easy word processing system to learn, but it is one of the most powerful. This program is noted for showing, on the video screen, text exactly as it will appear in print. Page break locations, the bane of Scripsit users, are indicated on the screen as a line of dashes. Almost unlimited formatting capabilities are available.

For systems equipped with the proper printers, Wordstar allows for micro-spaced justified text, as well as a full complement of special printer functions such as super- and subscripts, overstrike, alteration of character width, over-printing, boldface and emphasized print.

Text files are limited only by disk storage capacity. Files may be read into working text and portions of text being worked on may be saved to other files. Additionally, Wordstar has a "non-document" mode that is an excellent text editor for preparation of source code for assemblers, Pascal and other high-level languages.

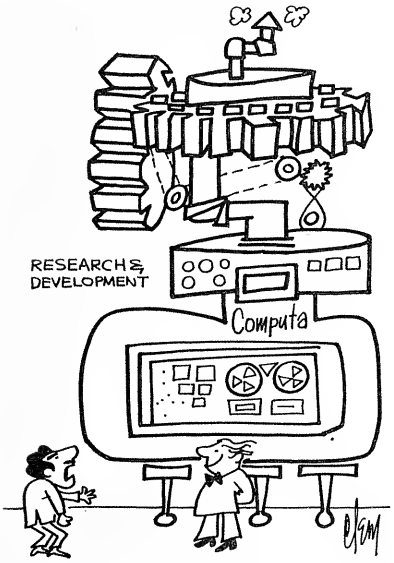
The Mailmerge option supplied with this package is designed to produce form letters, insertion (automatically) of additional files during printout, and user-supplied data into the printout. Mailmerge is a "must have" addition to Wordstar for business use that expects to generate any boiler plate text. The ability to include entire files allows you to create a multi-page document with only a few lines of command directives. I use this utility often to generate the inside address for letters as well as typing the address on the envelope for the letter. Another fine feature of Mailmerge is its ability to generate multi-column labels.

Wordstar, in common with many programs that operate under CP/M or MS-DOS, is supplied with an "install" program. This utility configures the program for the unique hardware you are using. Installing LDOS Wordstar is slightly different than a conventional CP/M installation. The install program, which is menu driven, lists options for running under LDOS, Newdos80 and DOSPLUS. Directions with the install section of the documentation state that only the LDOS version has been tested.

Installation for specific printers is also menu driven. One of the printers listed on the menu is the Radio Shack Daisy Wheel II printer, but the documentation states that this option is not functional. If that is true, it is hard to understand why this printer is shown on the install menu. The only other Radio Shack printer that is listed on the printer installation menu is the DWP-410 daisy wheel printer. Unfortunately, these are the only Tandy printers that may be installed automatically. Any other Radio Shack printer could be used, but installation would have to be done manually, following directions for custom printer installation in the documentation.

Another irksome problem with the installation procedure is a lack of sufficient disk space to hold the uninstalled version of Wordstar, two overlays, the install file and, finally, an installed version of the program. If you follow the directions given, the installation will blow up due to lack of sufficient disk space! It will be necessary to either kill some of the system files on the disk, such as BASIC, or run the install program without one of the Wordstar overlay files. This overlay file can be copied to the disk after completion of the install process.

During my testing of the product on a stock Model III, an NEC Spinwriter printer and running smal-LDOS as supplied, all major features of Wordstar were tested and found to work. I use Wordstar on a daily basis, both at work and at home writing articles. I have used Wordstar on many CP/M computers and a few running MS-DOS. The LDOS Wordstar combination of hardware and software produces the *slowest* implementation I have ever used. The



"Franklin, just how are we going to make that the industry standard?"

Reviews

problem seems to be twofold. First is the slow clock speed of the Model III, coupled with the LDOS overhead applied to Wordstar, and second is the limited memory available of Wordstar which results in far too frequent disk accesses, with the notoriously slow Radio Shack drives. This program would be improved quite a bit if it were running with a hard disk. Not only would the very slow disk access be improved but, also, the limit on file sizes for text would be expanded.

With a price of $400, this program is not inexpensive. If you *must* have Wordstar and do not want to opt for a CP/M modification, this will serve the purpose. I hope that Logical will offer the Spellstar spelling checker with this program in the future. Spellstar is a fast, fool-proof utility that completes the package. I would also hope that Logical Systems will offer a version of this for the Model 4 instead of only the Model III. On the Model 4, with its faster clock, the program will run somewhat quicker. Even better would be a version that utilizes the extra 64 kilobytes of memory of the Model 4 to provide for an increased buffer, thereby reducing the time-consuming disk I/O routines.

Harry Avant

PFS:File
Models III/4
Radio Shack Computer Centers and stores
Developed by Software Publishing Corporation
$124.95 Model 4 version #26-1518
$124.95 Model III/4III ver. #26-1515

What's Radio Shack's hottest selling data base management program? If your guess is Profile III Plus, you're about a year behind the times. Since the introduction of PFS File, the Profile program has nearly sunk from sight in the sales reports while PFS has risen to be the number four seller in the Radio Shack software line. The reason for the change in emphasis by Radio Shack is, I believe, clear. While Profile is a tremendously powerful program, it is also tremendously complicated to master (the manual runs 200+ pages). Radio Shack needed a program that could be set up by a business in a couple of hours rather than a couple of days. PFS was to meet that need.

Unlike Profile, which was so extensive that it required two diskettes for the Model I/III, PFS is so concise that it actually becomes resident in memory when the diskette is booted. Once booted, one removes the PFS diskette and builds the file on a TRSDOS diskette in drive 0. This is an advantage to the user because none of the diskette space is wasted on the mother program — just on the DOS.

The first thing that one notices about PFS is that the manual is well-written and quite simple to understand. This will come as quite a relief to those who suffered through the Profile manual. PFS allows as many as 50 fields per screen and 32 screens per file. It also allows "attachment pages" with each record. These are actually full-screen fields in case you need extensive notes added to any record(s).

The next thing that one notices is that it is very simple to define fields and set up screens using PFS. One is given a blank screen and merely places fields wherever one pleases. PFS does all the rest. You don't have to number your fields or define field indicators or decide which fields are to be right-justified. You needn't worry about plus or minus fields or write-protected fields or decimal fields. You don't have to know how to set up math calculations on your fields or between fields. You don't have to worry about these things because PFS doesn't have the ability to do them.

What, then, does PFS do if it doesn't allow for any of these important qualities? It is here that we must come to grips with the advantages and limitations of PFS. PFS is a store-and-fetch program which is simple to learn and simple to use. If you need the power and versatility of Profile, or have used Profile extensively, you will probably not be interested in this program. If you need to store information and search that information in various ways, but don't want to spend a good deal of time learning a complicated program, PFS is probably what you are looking for.

PFS's search routines are very good (in fact, I find them superior to those of Profile). The most sensible feature in the PFS search is that it searches from the end of the file to the beginning. This makes sense because we usually deal with recent data rather than older data. PFS does a binary search on the first field (the key field) but is limited to a sequential search on all other fields. This, of course, is a slower search routine. Like Profile, any record can be updated or deleted once it is found and that disk space is then reallocated.

Besides the lack of power mentioned above, PFS has some other problems — problems which are, in my estimation, serious enough to steer one away from the program entirely. First of all, the keyboarding is awkward. Profile moves to the next field when one presses enter or downarrow. In order to move to the next field in PFS, one must press shift and rightarrow — a very awkward, two-hand operation. Neither the break key nor the clear key are protected by the "double-press" operation available on every other

data base management program which I've used. Pressing either of these keys during screen input will make you very sad.

If you have a great many records to store and are using Profile, you can simply add external disks to your system or use a hard disk system. Your storage capacity then becomes nearly limitless. The reason, of course, is that Profile will simultaneously access as many disks as you have in your system. Not so with PFS. This is, in my estimation, the most serious drawback to PFS. I would even go so far as to say that it is the fatal drawback. Although you must have a two-disk system in order to backup your PFS records, the program will only read or write to drive 0. Therefore, the more complicated you make your program (remember that you are allowed up to 50 fields per screen and 32 screens per file plus attachment pages) the fewer records you will be able to store. The solution offered by the manual is the breaking up of your records into several diskettes and popping the diskettes in and out of drive 0 as you search or print reports — a rather weak solution, I would say.

Another weakness of this program is, in fact, a weakness in every data base management program — that of redesigning the screen without destroying existing records. PFS claims the ability to do this but warns you to be sure to have a backup of your records before attempting it. Once you have redesigned your screen, PFS will reorganize your files to fit into the new screen design. This reorganization, says the manual, "can take from 5 minutes to a few hours." That's cautious optimism if I've ever heard it.

PFS File is a limited data base management program which is aimed at the non computer-oriented business person who needs a basic (and easy-to-learn) way to store and retrieve records, but doesn't need to manipulate those files in any way other than searching by any of the fields and printing simple reports. The program has a number of limitations — the most serious being its inability to access more than one disk drive. The program, therefore, not only fails to use most of the computer's internal power, it also fails to use all of the computer's storage capacity — a shame.

Radio Shack is very proud of the disk protection on this program. They have made it essentially pirate-proof. As one Radio Shack employee told me, if the authors had spent as much time on the program as they did on the software protection, he would have been more impressed.

Michael J. McMorrow

VIP Writer
(Super Color Writer II)
Color Computer
Softlaw Corporation
9072 Lyndale Ave. So.
Minneapolis, MN 55420
(621) 881-2777
orders (800) 328-2737
ROM pack $59.95, 32K disk $59.95

"Word processing on the Color Computer? Ha! Don't make me laugh!" That was what went through my head when asked to review Super Color Writer II from Softlaw Corp. (formerly Nelson Software Systems). After just a couple of minutes with this fine software, I changed my mind completely. After spending more time with it, I think I'm in love.

To compare this word processor to Scripsit on the Models I or III would be unfair, but that is what I have been using for quite some time. Surprisingly, Super Color Writer is just as good, or better, in some respects. That really floored me. At $59.95, it is an even better buy.

The first thing that the new purchaser

will see is the screen display. It powers up in the 51x21 mode with 32K or more. The display is very readable and doesn't strain the eyes. The 64x24 mode is the highest I can use with my monitor. Either 21 or 24 lines may be used and the screen may be set to 32 (standard Color Computer display), 51, 64 or 85 columns. In all but the low-resolution mode, the lowercase characters have descenders and all have an extremely pleasant appearance. Unfortunately, owners of 16K machines will have to upgrade their machines to at least 32K in order to use the high-resolution features. With the price of RAM as low as it is, that should present no problem for most people. Even the keys click when pressed if the volume is turned up on the TV.

When first run, the program shows a pleasing display of the copyright information and then determines for itself how much memory you have. It is claimed that full use of a 64K machine is automatically accomplished. Since I have 32K, I couldn't verify this.

When entering text, a full complement of cursor motion controls is available. The arrow keys move the cursor in the indicated direction. Using various combinations of shift, clear, and an arrow key, you can place the cursor at the beginning or end of text, move ahead or back one page, move right or left one word (or line), and home the cursor to the top of the screen. User-positioned tabs may also be reached, but shift leftarrow was not a good choice of keys for this. Shift rightarrow would have been better.

The text buffer may be cleared from before or after the cursor, or completely, with the Before, After and Clear commands. Text mistakenly cleared may be restored with the "oops" command. What a very good thought and a delightful choice of words this is. Should you need additional memory, the text buffer may be extended by entering the low-resolution mode. This adds about 8K to the text buffer, but at the sacrifice of the beautiful display. Since files may be chain printed, this feature will probably not be needed unless editing is done on a BASIC program which, by the way, is more convenient than the editor supplied in the ROM. While editing, you may also perform a word count and determine the amount of memory remaining and the size of the file both in bytes and grans.

The editing features are very good. You may toggle on an insert mode wherein all text will be inserted at the cursor position and all other text moved to the right. I like this type rather than the character-by-character method of some other word processors. Although slower than some because of the 1MHz clock, the type-ahead makes it very workable.

If you think the "oops" command is useful, how about the "undo" command? If you have inserted text and change your mind, simply press break and your text will be as it was. This works with delete as well as insert. When in line insert, this works only on the current line.

The locate function is indeed powerful. In fact, I was pleasantly surprised at all the things this one can do. You may locate, locate and replace, or globally locate and replace. Most functions work identically, so we'll simply discuss the locate. Suppose you want to search for a string of characters. No problem, this software does it. Suppose you want a case independent search. Again, no problem. Perhaps you want to search for a string like "Johannssen," but you can't remember the exact spelling. Simply search for "Johan***en." Use the right-arrow where I have used the asterisk. This use of wild card search is powerful. If you are doing a global search and replace, and you think that you could have made an error, ask for the search and replace with verify. This will prompt you before executing the replacement with the cursor indicating where it will be made. Mercy, that's powerful.

You can also move text around by first marking it with block markers. Not only can you move it, you can also copy it. How often I've needed that ability and not had it. I hesitate to remember. Text marked with block markers may also be deleted. A prompt, "Are you sure?" is displayed as this feature will not respond to the undo or oops commands.

Super Color Writer II has a full complement of tape- and disk-access commands. All saves to disk are verified and error handling is good. Whenever an error occurs, a low beep comes from the TV. A directory of your disk may be easily obtained without leaving the program. It is also possible to save part, rather than all, of a text file. Because it is cursor-oriented, beginners might lose a file or two by having the cursor in the wrong place. This is overcome by having a prompt, "filename PART SAVED" appear whenever the cursor is not at the top of the file during a save. If the save would cause an overwrite of an existing file, the prompt "Are you sure" will be displayed and a response of Y will save the file. An extension of /SCW is automatically added to any file saved without extension. If you specify a different extension, it will then become the default extension. Files may be appended to the text in memory, renamed, and killed as well.

This program supports many functions to control the way the printout will be produced. Text may be justified, printed flush right or left, or centered.

Headers and footers may also be printed on each page. You may imbed non-printing text within a document by using comment markers. A hard space, which will be printed as a space, but not broken up by justification or placed on the next line, is implemented. No one wants a name to appear on two separate lines or have "Fort Worth" spread apart with additional spaces. This is a feature needed by all word processors but omitted from many. In addition, pages may be numbered. Hyphenation is difficult to implement on any word processor. Super Color Writer II handles this difficult chore adequately.

Printing is well done with this program. It is necessary to imbed your special printer codes within each document. I would have preferred that a system of configuration be used requiring the system to be set up only once. Once learned, these are easy to use, but it still would be better to create a file and have the program use it together with a standard code for each function. If you change printers, you will have to change the codes in all your documents. Although not the best way, this is workable and, through the use of programmable character codes, quite easy to get around. This way does allow more versatility, though. If your printer will allow it, you may superscript, subscript, underline, and backspace nearly anywhere in your text. There are many expensive word processors that don't support all of these abilities.

It is also possible to link disk files for continuous printing. If you write long documents, you'll find it hard to get along without it. Super Color Writer II handles this without difficulty. Tape files may also be linked together for printing. Since all files are stored as ASCII text, they don't have to have been created with Super Color Writer II in order to be used by it.

The documentation for this program is superb. In addition to explaining the program thoroughly, it also tutors you in the fundamentals of word processing. Terms that are new to most people are well explained in the text and in a glossary in the back of the manual. If anything, it explains too much rather than too little. If you're writing a program for public distribution, you can learn a lot by studying the way this manual is written. It is a truly professional job in all respects. It is well indexed and serves well as a reference after the program is learned. Command summaries or "cheat sheets" as I sometimes call them, are located at the back of the manual. There is even a section on how to get the most from your color TV when using it as a monitor.

The manual is 109 pages in length and packed with information you'll find missing in many other manuals. The binder is brown plastic, three-ring, with a pocket for the diskette. It is impressive, particularly when one looks in the front and sees that it is written by the programmer himself. Nothing is perfect, and this manual has a two-page errata sheet in the front. This explains where every known error is so that you may find and fix it before proceeding. Mr. Murphy was obviously there during the writing, but he didn't get the best of the authors.

Effective October 1, 1983, the name of this program was changed from Super Color Writer II to VIP Writer. In addition, a backup disk is included with the purchase. Nelson software assures me that they have a liberal policy regarding replacement of a damaged disk.

Word processing on the Color Computer? You bet! With the addition of one of the custom keyboards on the market, and a daisy wheel printer, you can have printed matter that's the equal of anybody's. You can also produce it with an ease that many ten thousand dollar machines can't match, and for only a fraction of that cost. I

enthusiastically recommend the purchase of Super Color Writer II by anyone who has a Color Computer and uses his typewriter more than once per month.

Charles P. Knight

TRS-80 Color Computer Assembly Language Programming
By William Barden, Jr.
Color Computer
Radio Shack #62-2077
$6.95

We often have to wait a long time for something very special. William Barden's new book, *TRS-80 Color Computer Assembly Language Programming* was definitely worth the wait. It offers a complete guide to programming using Radio Shack's Editor-Assembler-Debug package EDTASM+. His text gives the aspiring programmer a firm foundation in assembly language techniques. And, of course, it describes programming the Motorola 6809 microprocessor contained in the Color Computer.

Getting started with assembly language programming is not easy for most people. Higher level languages, like BASIC and Pascal, are designed to hide the machine's details from the programmer. With assembly language, however, you are working at the machine level of bits, bytes, and data registers. You must learn about the internal workings of your computer. William Barden's approach begins with real examples and leads the newcomer through a complete "first course" in assembly language programming. In doing so, he also shows how to use the EDTASM+ editor/assembler program, which forms the programmer's link with 6809 microprocessor instruction set. The experienced programmer will also find this book useful as a quick reference on Color Computer assembly language and EDTASM+ instructions.

The book is full of sample assembly language programs which make broad use of EDTASM+ capabilities. The EDTASM+ program is a very good choice for an editor/assembler to use on the Color Computer. Its many interactive features make assembly language programming easier to learn. EDTASM+ fully integrates the three phases of programming: entering and editing program text, assembling the program into machine instructions, and debugging. The ZBUG debugging monitor is an interactive, symbolic debugger. It allows reference to program lines by using labels instead of hexadecimal addresses.

The book presents assembly language programming in 28 chapters. Six appendices summarize 6809 instructions, EDTASM+ commands, decimal-hexadecimal conversions, and the ever-present ASCII code. The first 18 chapters provide coverage of assembly language for the 6809, addressing modes for the 6809, and usage of EDTASM+. The next three chapters cover interfacing assembly language programs to BASIC and passing parameters to and from assembly language programs. Two chapters discuss advanced addressing modes and how to utilize some of the BASIC ROM routines. You will have to consult other sources to extend your list of useful ROM routines. Two chapters introduce the reader to graphics and sound on the Color Computer. Again, more extensive information can be gleaned from magazines. The final chapter discusses the complete assembly language process and gives guidelines for program design, flowcharting, coding, debugging, and documenting larger assembly language programs.

William Barden's light, unpretentious style should be familiar to Radio Shack computer owners. He has written a number of introductory and intermediate texts on programming most of Radio Shack's computer products. His book *Color Computer Graphics* is of special interest to Color Computer owners. It describes graphics programming in Color BASIC and Extended Color BASIC. He frequently refers to this work throughout the assembly language text.

A "Key Chart" indexes text coverage at the beginning of each chapter. All of the assembler mnemonics and instructions are listed, as well as the ZBUG commands and general topics. Items covered in previous chapters are crossed through. Topics to be covered in the current chapter are set in bold type. This feature allows the reader to quickly determine where he is in the overall scheme of things. I felt a sense of accomplishment to see the topics checked off as I proceeded from chapter to chapter. Having a key chart makes finding topics somewhat easier. You know whether to page forward or back to find a particular topic.

The discussion of assembly language programming is very easy to follow. Each topic is enhanced by clear examples and assembly program listings. The reader can enter each example and run the program with complete step-by-step instructions. Many of the examples contain questions for further study. Each chapter closes with a review of topics covered and refers

to other parts of the text for additional information.

Barden's wit is sometimes very sharp. You can be hit unexpectedly with a quick jab. After wading through several pages of description on rotates, shifts and multiplies, you suddenly encounter the BRAAB instruction. Barden informs us that this seldom-used instruction stands for "Branch And Bomb." This outlandish mnemonic is not really a 6809 code at all, but a creation of Barden's imagination. Other fake gems include BTU (Branch if Tuesday) and BOC (Branch Occasionally). It turns out that some unlikely instructions such as BRN (Branch Never) are actually legitimate, so you have to be sharp to catch the zingers.

A few features of the book deserve special mention. Barden uses pictures to illustrate how a program references memory. These pictures enhance and clarify the discussion. He also illustrates how to use index registers for addressing a table of values. I was also impressed with the clear explanation of conditional branch operations. He explains particularly well the subtle differences between comparing signed and unsigned numbers.

The sample programs illustrate the power of assembly language programming. One of the best is a bubble sort which arranges all the characters on the screen into alphabetical order. It has a dazzling effect as letters scurry across the screen to find their proper locations. Another excellent sample program is a sound routine which forms the basis of a rudimentary music synthesizer.

With all these pluses, the book has a few minuses as well. Sometimes, the text descriptions don't match the program listings exactly. For example, the text discusses loading the B register with FF (hex), while the program has the line LDB#$55. Most errors occur with isolated lines of code typeset in the text.

A few promised items are missing from the text. For example, EDTASM+ pseudo-ops are not in Appendix III as mentioned in the text. Indentation of loops referred to in the text are not provided in the source code. Figures 15B, C, D are missing. All in all, not too bad for the first printing of a 294-page programming text.

An introductory text should provide a path for the reader to explore. However, this book lacks references to works outside Barden's own. Although he mentions his own book on Color Computer graphics, he doesn't refer to other books which can provide more information on related topics. For example, the text doesn't really get into interrupts or programming the I/O devices in the Color Computer. General introductory I/O interfacing and programming for the 6809 can be found in Michael Andrews book *Programming Microprocessor Interfaces for Control and Instrumentation*, published for Motorola by Prentice-Hall. The bible of 6809 programming is Lance Leventhal's *6809 Assembly Language Programming*, which gives an in-depth coverage of 6809 programming and software development. Color Computer graphics is covered from the assembly language point of view in Don and Kurt Inman's *Assembly Language Graphics for the TRS-80 Color Computer*.

I am quite pleased with this latest work from William Barden. I now have a solid book to recommend to those beginning Color Computer programmers who want an introduction to assembly language. The combination of Barden's text and Radio Shack's EDTASM+ is a sure winner.

Stuart Hawkinson

How To Enter Our Listings

Our program listings come directly from the submissions of our authors We do not edit them at all (that's why you sometimes see spelling errors in them) We run all submissions and make sure that they do work

To enter one of the listings given make sure you have the type of computer specified and all necessary programs, operating systems, or hardware that the program uses. Type in the program exactly as it appears in the magazine. Be extra careful so you do not confuse 0 (zero) with O or 1 (one) with I or L. Save the program to tape or disk **before** running it. On long programs it is wise to save it as you go along, thus protecting yourself from having to re-enter the whole program if the lights go out

Here are some tips to help you catch errors that you may have made in typing If you get an out of data error, the problem lies in the DATA statements, rarely in the READ line that the computer refers to Check all DATA lines to see that they are correct and that no commas or values are missing It might be useful to print each variable after it is read, that way you can follow the computer as it goes through the data Just insert a :PRINT **variable** right after the READ **variable** command.

Many of our authors use a linefeed, or downarrow, in their programs If you see lines of code that have many blank spaces and then they begin again on the next line with more code, a linefeed was used Even if you don't use them, the program will run but the video display may be messed up

You will find the TRON command helpful in following the program's logic By turning the trace command on, TRON, you can see what lines are being executed by the program It is very useful in catching GOTO or GOSUB errors and incorrect references to linenumbers. Don't worry about video formatting when the trace is on, it will be quite messy.

If you find yourself getting TM or type mismatch errors, check carefully the use of the $ symbol. Also look at the beginning of the program to see if you correctly entered the DEFINT or DEFSTR statements

Function call errors usually occur when a variable has a value that is not allowed Check all variables that are being used by the function, one of them probably has the wrong value

If after all that, you can't get it to run, send us a paper listing of your program, what systems you are running it on, and carefully document the error you are getting. We will do what we can to find the flaw It is very difficult for us to try to help you debug errors over the phone. Check Letters and Notes, etc in the next few issues for updates or conversions. Many times a reader will tell how to embellish a previously published program

Bulletin board

This bulletin board space is available free to individuals with single or unusual items for sale or trade. Basic Computing reserves the right to reject any commercial advertising in this section and suggests using our display advertising for that purpose.

These notices are free of charge and will be printed one time only on a space available basis. Notices will be accepted from individuals or bona fide computer user clubs only. All these unclassified announcements must be typed, contain 75 words or less and include complete name and address information.

Snohomish Users Group meets the third Tuesday of each month at the Marysville library, 7-10 PM. Includes all models, contact J. Emge, P.O. Box 1323, Stanwood, WA 98292 or call (206) 653-6706 for more information.

CAPATUG: The Capitol Area TRS-80 Users Group meets the first Thursday of each month at the Fairview Township Fire House in New Cumberland, PA. For more information call the bulletin board (717) 774-6543 or write to CAPATUG, 340 Lewisberry Rd., New Cumberland, PA 17070.

Model II Card Reader for sale. Chatsworth Data MR 500 mark sense card reader with interface for Model II, 12, or 16. Best offer. Contact Jerry MyCue, Box 377, McQueeney, TX 78123, or call (512) 834-4292.

Micromint Expansion Interface for TRS-80 Model I. For sale at $275 postpaid. Used one year, never a disk read error, has printer port. Contact Fraser Smithson, 6235 Pontiac Tr. W., Bloomfield, MI 48033 or call (313) 682-1643.

32K Color Computer for sale. Includes Extended BASIC, two joysticks, disk drive controller, Macrotorn Premium keyboard, software. Only $200, will ship COD with a small deposit. Contact Warren Morrisett, 920 Palisades Ave., Union City, NJ 07087 or call (201) 866-0003.

Wanted: Model I expansion interface and disk drives. Contact Ron Goelz, 1111 S. 16th St., Escanaba, MI 49829 or call (906) 786-3941.

Bullet-80 BBS is located in Harleysville, PA. The system has many advanced features including user-activated auto log-ons, complete uploading and downloading, nine different message centers and more. Modem call (215) 256-6336, sysop is Dave Cardy.

Model I for sale. Includes interface, 48K, 5/8 doubler, keyboard pad, two disks, software, recorder. Priced at $1080, modem for $58, voice synthesizer for $55, free shipping. Contact Bob Green, 1315 Rustic Ridge Dr. N.E., Atlanta, GA 30319.

Model I for sale. Includes two disk drives, all cables, Newdos, TRSDOS. Priced at $700. Contact Chris Terrasi, 5633 52nd St. S.E., Grand Rapids, MI 49508 or call (616) 698-6030.

Model I for sale. Includes expansion interface, two disk drives, numeric pad, double density board, software. Priced at $1000. Contact Robert L. Sawall, 1744 So. 54th St., West Milwaukee, WI 53214 or call (414) 327-1174.

Model I for sale. Includes 48K, one disk drive, double density board, expansion interface. Original software (with documentation) includes Scripsit, VisiCalc, DOSPLUS 3.4, Profile, Accounts Receivable, General Ledger, games and more. Priced at $1200, the software alone is worth over $800. Contact Dick Weerts, 4710 Valley View, Columbus, NE 68601, or call (402) 564-2170.

For immediate release

Computer interfacing projects

Four-outlet power filter


Computer cart

Computer Interfacing Projects

TRS-80 Models I, III, & Color Computer Interfacing Projects (ISBN 0-672-22009-1) by William Barden, Jr., gives techniques for translating real-world events into computer language. The $14.95 paperback has schematics, illustrations, diagrams, equipment suggestions, and many specific examples. The text is available at local bookstores or contact Howard W. Sams & Co., Inc., 4300 West 62nd St., Indianapolis, IN 46268 (317) 298-5400.

COBOL Fundamentals

Radio Shack has a new educational software package designed to teach the fundamentals of COBOL programming. The material is designed for secondary and college students and adults. The material can be used in a classroom setting or as a self-teaching program. The software, which is available for the Models II, 12, or 16, sells for $49.95. Its use requires the COBOL Development System (#26-4703, $299). For more information contact your local Computer Center or dealer.

Four-outlet Power Filter

The Wire Tree™ is a four-outlet filtered power source which mounts easily on the computer work station. It offers continuous protection through its surge-limiting solid-state circuitry that is built-in to its four grounded outlets. The circuitry absorbs any dangerous spike energy before it reaches the computer by providing an extremely rapid decrease in circuit impedance.

The device filters out transverse and common RFI noise. Control of the total system power is from a single, illuminated on-off switch which is recessed to prevent accidental shut off. The Wire Tree™ is protected by an 8 ampere fuse and has a one year warranty. For more information contact NETWORX, 203 Harrison Place, Brooklyn, NY 11237 (212) 821-7555.

PC-2 Forecasting

PocketInfo Corporation has a graphics trend-analysis and forecasting program for the Radio Shack PC-2 and Sharp PC-1500 computers. It can provide moving average, exponential moving average and least squares analysis of your data. Data is saved in a spreadsheet-like file and can be graphed in multiple colors. The graphs show both the original data and the trend or forecast line. Up to three trend lines can be plotted on each graph. The program sells for $29.95 on cassette tape. For more information and a free catalog of over 40 other programs for the Radio Shack Pocket and Model 100 computers, contact PocketInfo Corp., P.O. Box 152, Beaverton, OR 97075 (503) 649-8145.

Computer Cart

The EC-15 computer cart has an


Media safe

adjustable top shelf and work station as well as a single, neon-lighted switch which conveniently turns the complete computer system on and off. The beige colored table features heavy gauge steel construction, with one inch tubular legs on four inch casters, two with locking brakes. For more information about the EC-15, or a copy of the Bretford catalog, contact Bretford Manufacturing Inc., 9715 Soreng Avenue, Schiller Park, IL 60176 (312) 678-2545.

Media Safe

The Model 5750 Computer Software Safe, from Sentry Safe Company, is designed to protect diskettes from fire and theft. The safe will hold forty eight-inch diskettes.

In actual tests, the safe was exposed to temperatures up to 1700° F. and the interior remained below 125° F. There was no loss of data. The safe can hold 5¼-inch diskettes, microfiche, and tapes as well. It has a 4-inch locking bolt, three-number combination, and comes with a three year warranty. For more details contact the distributor, Value-tique Inc., Dept EDP-58, P.O. Box "B", Leonia, NJ 07605.

Color Monitor

The CB-141 color composite monitor, from Roland DG, is designed to give full color graphics at an affordable price. The unit


Color monitor

measures 14 inches deep and its display size is 9¾ inches horizontal by 6¾ inches vertical. A standard RCA pin connector easily attaches to the back of the monitor making it adaptable to most computer systems.

The CB-141 features high clarity, built-in speaker and audio amplifier, and a headphone jack. It can also be used with a video recorder for home or professional use. Retail price is $399. For more information contact Roland DG, 7200 Dominion Circle, Los Angeles, CA 90040 (213) 685-5141.

Intelligent Disks

A new line of high-quality disks, called Intelligent Disks™, with unique sensors which can warn users when a disk has been exposed to excessive humidity or temperature, has been introduced by Omni Resources, Inc.

The disks can help detect conditions that may lead to a loss of data. One sensor changes color at 120 degrees F. and the other gradually changes from blue to red in a range of eight to 80 percent humidity. Suggested retail price for a box of ten SSDD disks is $52.50 and they are available from retailers who carry the Omni line of materials. For more information contact Omni Resources, Inc., 50 Howe Avenue, Milbury, MA 01527.

TRS-80 Data Files

Wayne Green Publications Group has announced a new book entitled *Introduction to TRS-80 Data Files*, a beginner's guide to writing a database manager. Included in the text is disk containing all programs used in the text. The author, John D. Adams, covers both sequential and random access methods and develops several mailing list

Intelligent disks

programs. Instructions are given for modifying the programs for either Model I or Model III use. The package sells for $24.95 and is available from WGBooks, Peterborough, NH 03458.

Multiple Client Accounting

CASP is a client accounting software package for the public accounting profession. THe system is written in COBOL and runs on the Models II, 12, and 16. CASP includes a multiple-client general ledger with integrated payroll. User-defined report generation permits printing of letter quality client financial statements with full disclosure, formatted according to the accountant's specifications. Complete payroll reporting includes quarterly 941 and state unemployment reports.

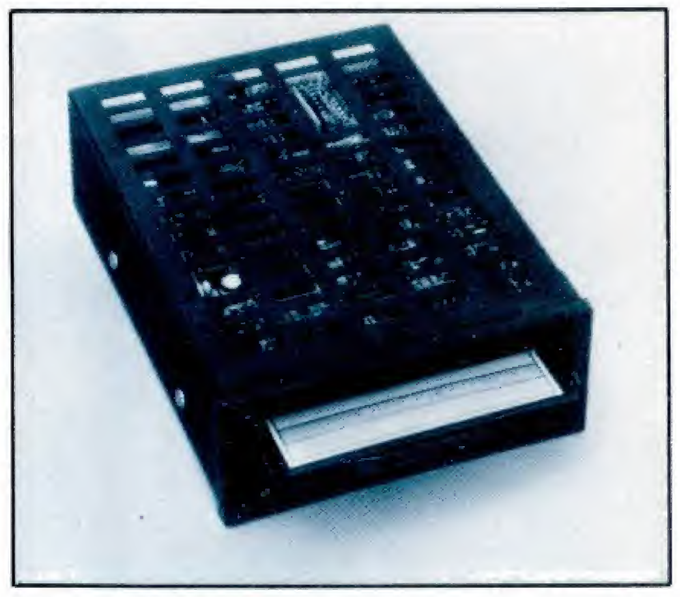
Price is $1,000. The program may be upgraded to a multi-user Model 16 system. A 45-day trial diskette with documentation is $50 (COBOL run-time package is required), applied against purchase. For more information contact David Ray, CPA, 1301 Northwest Highway, Suite 210, Garland, TX 75041 (214) 840-9531.

Four-Color Ballpoint Pen Plotter

Ricoh of America, Inc. has introduced the GP-1, a new four-color ballpoint pen plotter for business and personal computers. It has a printing speed of up to 120 mm per second and can print in black, red, blue, and green ballpoint or felt-tip pen type. It features 26 plotter

Four-color plotter

Model 100 software



Micro disk drive

commands and an oil pen that lasts 1000 meters.

The GP-1 can use a variety of paper types, including 10-inch wide, sprocket-driven roll paper, 8x11-inch overhead transparencies and 8x11-inch sheet paper. For more information contact Ricoh of America, Inc., 20 Gloria Lane, Fairfield, NJ 07006 (201) 575-9550.

The Enigma Machine

Powersoft's new program, Enigma, is a text encryption program for the Models I/III/4III. By using a two-key encryption method, it can be used to exchange encrypted data from one TRS-80 user to another. The program is useful to those seeking data security as well as single users interested in cryptology. The program comes with a $250 reward for the first person to decode Powersoft's encrypted message that comes with it. For more information contact Powersoft, 11500 Stemmons Fwy., Suite 125, Dallas, TX 75229 (214) 484-2976.

Cassette Backup System

The Beta-DTS allows for automatic backup of diskette material to cassette. One megabyte of material can be saved on one C-60 cassette. The device will give exact and complete copy of a diskette's contents. Retrieval of a diskette from taped media is also automatic, including formatting. The Beta-DTS comes with a self-booting operating system diskette and manual. Minimum hardware requirements are a 32K TRS-80 Model I or III, cassette recorder, and one disk drive. The Beta-DTS, which sells for $109.95 (Model I), $113.95 (Model III), plus $3.50 s/h, can save money by avoiding having to purchase a second disk drive. For more information contact Beta Enterprises, Inc., 14049 Settlement Acres Dr., Cleveland, OH 44142.

Model 100 Software

American Micro Products, Inc. has introduced a full line of Model 100 software. They have programs for electronic spreadsheets with various templates, a portfolio analysis program for stock analysis that can automatically retrieve data from the Dow Jones News Retrieval Service, an income property analysis program, a statistical curve fitting and plotting program, a histogram and bar chart program, a generalized equation solving program, and a RPN calculator program for those who prefer to use the RPN method for calculations. The company also offers a FORTH language for use on the Model 100 that implements the 79 standard of the assembly-like programming language. The packages are priced from $29.95 to $99.95. For more information contact American Micro Products, 705 North Bowser, Richardson, TX 75081 (214) 238-1815.

Inventory Control

Retail Inventory Control Professional System manages up to 3500 stock items with a fast disk sort, rapid execution, writes invoices, discount invoices, orders stock, and posts sales. It also features search, editing, and space recovery on terminated stock items. Printed reports consist of complete stock listings, listings by prefix, out of stock items, below reorder line, and listings by vendor. The program, for the Models I/III/4III, is priced at $148. For more information contact Freeman Software, 334 Fieldside Dr., Garland, TX 75043 (214) 270-0625.

CoCo Database Manager

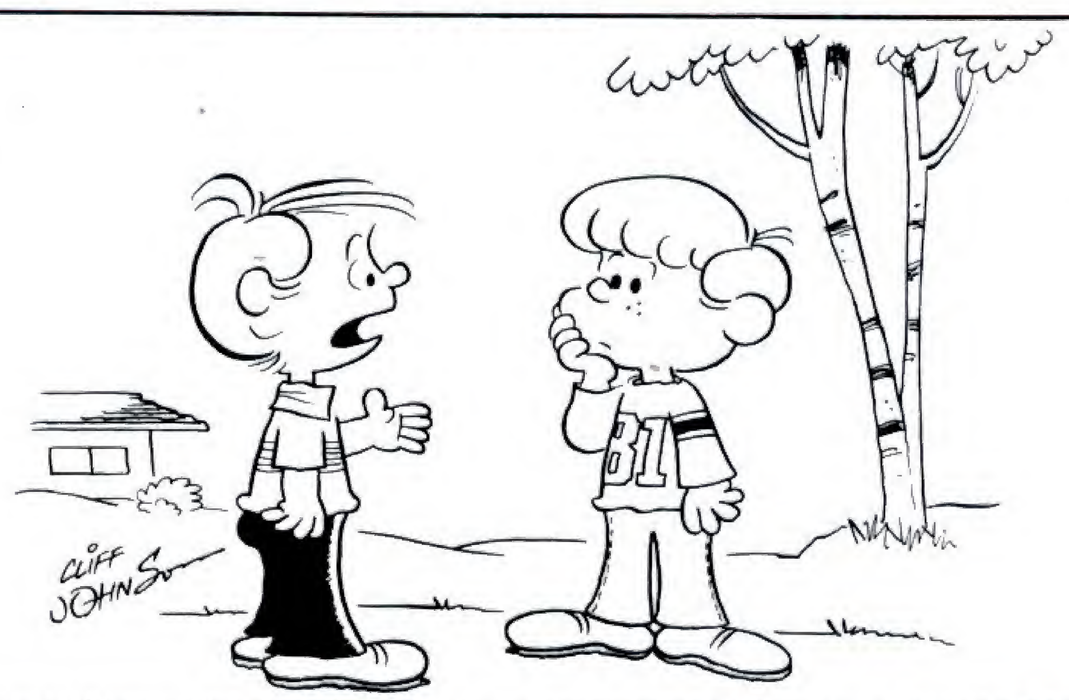
Elite-File™ is a relational database manager, with full editing and report generation, for 32K Color Computers with Disk BASIC. The program can handle up to 2,000 characters per record, up to 4,000 records per file, up to 16 files open at one time, up to 255 characters per field name, and up to 255 fields per record. The program is menu-driven and uses single keystroke commands. Records are edited with a full-screen, type-over editor. Sorting can be on any field, in ascending or descending order. Math operations and formulas are supported and totals can be printed. Field definitions can be copied from file to file. Data files can span disk drives. Retail price is $74.50 plus $2 s/h. For more information contact Elite Software, P.O. Box 11224, Pittsburgh, PA 15238.

Micro Disk Drive

MPI has introduced two new 3¼-inch microfloppy disk drives that use a standard 5¼-inch interface. The Model 321 is single sided one-half megabyte and the Model 322 is a double sided one megabyte floppy drive. Both machines are small (1.625"H x 4.0"W x 5.5"D), have a mean time between failure of 12,000 hours, head access time of six milliseconds, and are plug compatible with standard double-sided, double-density, 96tpi, 5¼-inch drives. Media format is 80 tracks per side, 140 tpi. The Model 321 sells for $155 and the Model 322 sells for $190. For more information contact contact local MPI dealerships.

Advertiser index

□ Contact these advertisers directly.



"Forget that stuff about the stork. Actually, you're programmed into a computer and **then** you're born!"

SWIG©

SOFTWARE WRITERS INTERNATIONAL GUILD

THE LARGEST PAID MEMBERSHIP PROGRAMMERS GUILD - OVER 5,000 MEMBERS WORLDWIDE!!

SCHEDULED SWIG ACTIVITIES & MEMBERSHIP BENEFITS

(1) $10,000 PROGRAMMING CONTEST (Members only)

(2) NATIONAL COMPUTER WEEK (May 11- May 20, 1984)

(3) ANNUAL CONFERENCE AND SOFTWARE AWARDS CEREMONY (During National Computer Week)

(4) CONSULTANT REGISTRY (With computer store referral system for customized software)

(5) JOB PLACEMENT SERVICE (Free to individual members, fixed maximum fee to companies)

(6) FREE SEMINARS & MEETINGS LOCALLY

(7) SOFTWARE LIBRARY LENDING & EXCHANGE SERVICE (Professional quality assemblers, utilities, games, etc.)

(8) SOFTWARE LOCATION SERVICE (For companies & individuals-if it exists, **SWIG** will find it. If not, see #9)

(9) SOFTWARE DEVELOPMENT SERVICE (From novice to scientist, **SWIG** members can work on any project-from applications to games to R&D)

(10) LEGAL SERVICE

(11) AGENT (**SWIG** can represent you in sales to software publishers)

(12) 24 HOUR - 7 DAY BULLETIN BOARD SYSTEM (BBS) ACCESSIBLE BY COMPUTER FREE

(13) AND MORE!!!!

MEMBERSHIP APPLICATION FOR SOFTWARE WRITERS INTERNATIONAL GUILD

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

PHONE # ( ) _____

- CLASSIFICATION:

  ☐ NOVICE ☐ BEGINNER TO ADVANCED

  ☐ ADVANCED WITH ON THE JOB EXPERIENCE ☐ RESEARCH/SCIENTIST

- WHAT EQUIPMENT DO YOU HAVE EXPERIENCE WITH &/OR ACCESS TO &/OR PLAN TO BUY?

  ☐ MAINFRAME ☐ MINI ☐ MICRO ☐ DESIGN/R&D

  BRAND NAME(S): ☐ IBM ☐ XEROX ☐ APPLE ☐ TI

  ☐ COMMODORE ☐ RADIO SHACK ☐ ATARI ☐ OSBORNE

  ☐ TIMEX/SINCLAIR ☐ NORTH STAR ☐ HEWLETT PACKARD

  ☐ OTHER _____

- AREAS OF INTEREST:

  ☐ DATA PROCESSING ☐ BUSINESS APPLICATIONS ☐ GRAPHICS

  ☐ LEGAL ☐ VOICE ☐ MEDICAL ☐ APPLIANCE (HOME) CONTROL

  ☐ ROBOTICS ☐ GAMES ☐ MUSIC ☐ R&D ☐ OTHER _____

- MEMBERSHIP ACTIVITIES AND SERVICES OF INTEREST:

  READ THE LIST ON THE LEFT AND CIRCLE THE NUMBERS BELOW THAT APPLY.

  1 2 3 4 5 6 7 8 9 10 11 12

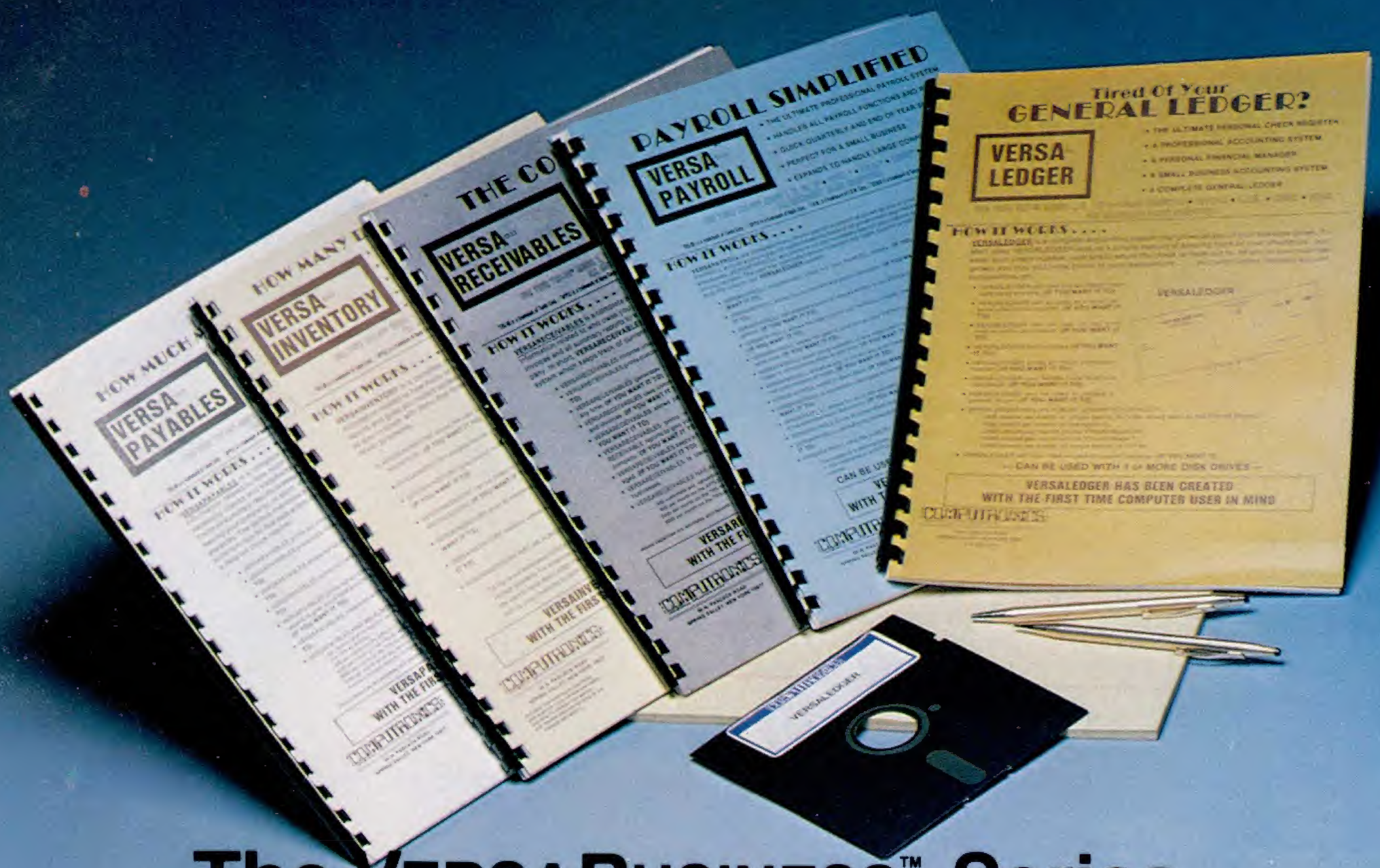
☐ I HAVE ENCLOSED $20 ANNUAL MEMBERSHIP FEE ☐ CK ☐ MO
(MAKE CHECK PAYABLE TO: **SWIG**)

RETURN TO: **SWIG** #71
P.O. BOX 87
STONY POINT, NEW YORK 10980
(914) 354-5585

SWIG© SOFTWARE WRITERS INTERNATIONAL GUILD

Introducing the Most Powerful Business Software Ever!

TRS-80™ (Model I, II, III, or 16) • APPLE™ • IBM™ • OSBORNE™ • CP/M™ • KAYPRO™



The VERSABUSINESS™ Series

Each **VERSABUSINESS** module can be purchased and used independently, or can be linked in any combination to form a complete, coordinated business system.

VERSARECEIVABLES™ $99.95

VERSARECEIVABLES™ is a complete menu-driven accounts receivable, invoicing, and monthly statement-generating system. It keeps track of all information related to who owes you or your company money, and can provide automatic billing for past due accounts. VERSARECEIVABLES™ prints all necessary statements, invoices, and summary reports and can be linked with VERSALEDGER II™ and VERSAINVENTORY™.

VERSAPAYABLES™ $99.95

VERSAPAYABLES™ is designed to keep track of current and aged payables, keeping you in touch with all information regarding how much money your company owes, and to whom. VERSAPAYABLES™ maintains a complete record on each vendor, prints checks, check registers, vouchers, transaction reports, aged payables reports, vendor reports, and more. With VERSAPAYABLES™, you can even let your computer automatically select which vouchers are to be paid.

VERSAPAYROLL™ $99.95

VERSAPAYROLL™ is a powerful and sophisticated, but easy to use payroll system that keeps track of all government-required payroll information. Complete employee records are maintained, and all necessary payroll calculations are performed automatically, with totals displayed on screen for operator approval. A payroll can be run totally, automatically, or the operator can intervene to prevent a check from being printed, or to alter information on it. If desired, totals may be posted to the VERSALEDGER II™ system.

VERSAINVENTORY™ $99.95

VERSAINVENTORY™ is a complete inventory control system that gives you instant access to data on any item. VERSAINVENTORY™ keeps track of all information related to what items are in stock, out of stock, on backorder, etc., stores sales and pricing data, alerts you when an item falls below a preset reorder point, and allows you to enter and print invoices directly or to link with the VERSARECEIVABLES™ system. VERSAINVENTORY™ prints all needed inventory listings, reports of items below reorder point, inventory value reports, period and year-to-date sales reports, price lists, inventory checklists, etc.

VERSALEDGER II™ $149.95

VERSALEDGER II™ is a complete accounting system that grows as your business grows. VERSALEDGER II™ can be used as a simple personal checkbook register, expanded to a small business bookkeeping system or developed into a large corporate general ledger system **without any additional software.**

- VERSALEDGER II™ gives you almost unlimited storage capacity (300 to 10,000 entries per month, depending on the system),
- stores all check and general ledger information forever,
- prints tractor-feed checks,
- handles multiple checkbooks and general ledgers,
- prints 17 customized accounting reports including check registers, balance sheets, income statements, transaction reports, account listings, etc.

VERSALEDGER II™ comes with a professionally-written 160 page manual designed for first-time users. The VERSALEDGER II™ manual will help you become quickly familiar with VERSALEDGER II™, using complete sample data files supplied on diskette and more than 50 pages of sample printouts.

SATISFACTION GUARANTEED!

Every **VERSABUSINESS™** module is guaranteed to outperform all other competitive systems, and at a fraction of their cost. If you are not satisfied with any **VERSABUSINESS™** module, you may return it within 30 days for a refund. Manuals for any **VERSABUSINESS™** module may be purchased for $25 each, credited toward a later purchase of that module.
All CP/M-based Computers must be equipped with Microsoft BASIC (MBASIC or BASIC-80)

To Order:
Write or call Toll-free (800) 431-2818
(N.Y.S. residents call 914-425-1535)

* add $3 for shipping in UPS areas
* add $4 for C.O.D. or non-UPS areas
* add $5 to CANADA or MEXICO
* add proper postage elsewhere

VISA AMERICAN EXPRESS

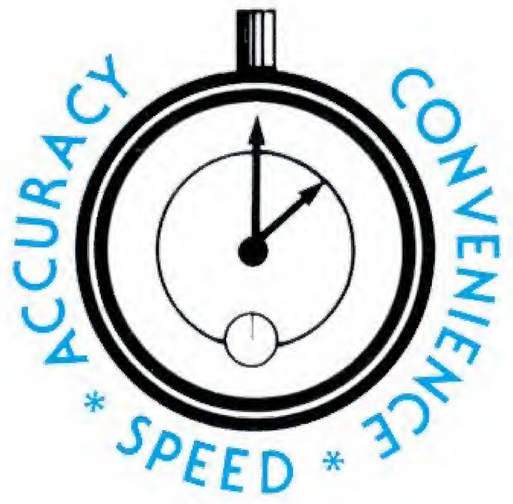
DEALER INQUIRIES WELCOME

All prices and specifications subject to change / Delivery subject to availability.

COMPUTRONICS INC

50 N. PASCACK ROAD, SPRING VALLEY, N.Y. 10977

CLONE I



CLONE III

CURSED.....! ROTTEN.....! COMPUTER...!

(error)

CRC doesn't have to stand for Cursed Rotten Computer if you have Clone. One of Clone's most powerful features is it's ability to recover CRC errors. Did you know that 95% of these errors can be recovered? Clone gives you a retry key which allows you to continuously reread a sector. Clone won't give up! Most backup utilities do!

Clone is fast! The same Clone program will automatically make backups of TRSDOS.... LDOS....NEWDOS....MULTIDOS....DOSPLUS....ATARI 400....ATARI 800....MODEL 1....MODEL 3 .....MODEL 4.....TRS-80 COLOR COMPUTER.....AND MORE. More impressively Clone will format and verify in one step, automatically, and all within 1 minute and 25 seconds.

CLONE CAN SAVE NEARLY 15 MINUTES FOR EACH 5 DISKS THAT YOU BACK UP ! ! !

Using Clone for your duplication needs will give you the time to back up each disk twice and the security of knowing that your disk has been freshly formatted and verified. CURSE YOUR ROTTEN COMPUTER NO LONGER!!

HARDWARE REQUIREMENTS

CLONE I OR CLONE III
TRS-80 MODEL III, IV OR
TRS-80 MODEL I WITH LN DOUBLER OR PERCOM DOUBLER ONLY
2 DISK DRIVES OR MORE
32K RAM OR MORE

NOTE: A MODEL I WITH SINGLE DENSITY MAY ALSO BE USED, HOWEVER YOU WILL ONLY BE ABLE TO COPY STANDARD MODEL I SINGLE DENSITY DISKS!

ORDER INFORMATION

$79.97

CLONE I for TRS-80 MODEL I
CLONE III for TRS-80 MODEL III or IV
For VISA or MASTER CARD Orders
CALL NOW - TOLL FREE (800) 824-7888
Ask for operator # 797
Hawaii & Alaska (800) 824-7919
For more information call (818) 367-0887

Add $3.03 Postage & Handling
California Residents add sales tax

(TRS-80 Color Computer and TRS-80 are trademarks of Tandy Corp. Atari is a trademark of Atari Inc. Clone I and Clone III are trademarks of Pegasus Research. LN Doubler is a trademark of LNW Inc. Percom is a trademark of Percom Data Corp.)

GIBBERMAN ENTERPRISES

Clone I and Clone III are available exclusively through Gibberman Enterprises, authorized dealer for Pegasus Research.

#72

Gibberman Enterprises - 13000 San Fernando Rd. - No. 5 - Sylmar, CA 91342

PEGASUS RESEARCH

An Affordable Alternative

smal-LDOS

VERSION 5.1
THE TRS-80™ OPERATING SYSTEM
MODEL I AND III

First Time Available to the Public Is This Unbelievably Priced Sub-Set* of the Renowned LDOS Operating System.

Order Yours Today for Only **$59.00** plus $3.00 Shipping & Handling

* Note: smal-LDOS does not contain all the features or utilities of the original "FULL LDOS" system.

Model-I smal-LDOS provided on 35 track single density media.
Model-III smal-LDOS provided on 40 track double density media.
smal-LDOS can be provided on special media configurations at an additional charge.
Prices & Specifications are subject to change without notice.
Although not required, LSI recommends two or more drives when using smal-LDOS.

DEALER INQUIRIES WELCOME. smal-LDOS is a product of LSI. TRS-80 is a trademark of Tandy Corp.

LOGICAL SYSTEMS INC.

Logical Systems, Inc. 8970 N. 55th P.O. Box 23956 Milwaukee, WI 53223 (414) 355-5454